UNIVERSIDAD PERUANA UNIÓN

FACULTAD DE INGENIERÍA Y ARQUITECTURA

Escuela Profesional de Ingeniería de Sistemas

Implementación de un sistema de detección y prevención de intrusos (IDS/IPS), basado en la norma ISO 27001, para el monitoreo perimetral de la seguridad informática, en la red de la Universidad Peruana Unión – Filial Juliaca

Por:

Yony Coyla Jarita

Asesor:

Ing. Jorge Eddy Otazu Luque

Juliaca, mayo de 2019

DECLARACION JURADA
DE AUTORIA DEL INFORME DE TESIS

Ing. Jorge Eddy Otazu Luque, de la Facultad de Ingeniería y Arquitectura, Escuela Profesional de Ingeniería de Sistemas, de la Universidad Peruana Unión.

DECLARO:

Que el presente informe de investigación titulado: "IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN Y PREVENCIÓN DE INTRUSOS (IDS/IPS), BASADO EN LA NORMA ISO 27001, PARA EL MONITOREO PERIMETRAL DE LA SEGURIDAD INFORMÁTICA, EN LA RED DE LA UNIVERSIDAD PERUANA UNIÓN – FILIAL JULIACA" constituye la memoria que presenta el bachiller Yony Coyla Jarita para aspirar al título Profesional de Ingeniero de Sistemas ha sido realizada en la Universidad Peruana Unión bajo mi dirección.

Las opiniones y declaraciones en este informe son de entera responsabilidad del autor, sin comprometer a la institución.

Y estando de acuerdo, firmo la presente declaración en Juliaca a los veintisiete días del mes de mayo del año dos mil diecinueve.

Ing. Jorge Eddy Otazu Luque

Implementación de un sistema de detección y prevención de intrusos (IDS/IPS), basado en la norma ISO 27001, para el monitoreo perimetral de la seguridad informática, en la red de la Universidad Peruana Unión – Filial Juliaca

TESIS

Presentada para optar el título profesional de Ingeniero de Sistemas

JURADO CALIFICADOR

Mg. Lennin Henry Centurión Julca
Presidente

Mg. Nilton Omar Santillan Aching
Secretario

Ing. Angel Rosendo Condori Coaquira
Vocal

Ing. Benazir Francis Herrera Yucra
Vocal

Ing. Jorge Eddy Otazu Luque
Asesor

Juliaca, 27 de mayo de 2019

DEDICATORIA

En primer lugar, a Dios por haberme guiado por el camino de la felicidad hasta ahora; en segundo lugar, a cada uno de los que son parte de mi familia a mi padre Roberto Coila Pari, mi madre, Valeriana Jarita Charca y no menos importante; a mis hermanos, a todos mis tíos y a mis amigos más cercanos, por siempre haberme dado su fuerza y apoyo incondicional que me han ayudado y llevado hasta donde estoy ahora.

AGRADECIMIENTOS

Por grandes razones, agradezco primero a dios, quien me dio la oportunidad de la vida y por ende de mi actual éxito, en segundo lugar, a mis padres, quienes me apoyaron en todo lo indispensable, al Ing. Jorge Eddy Otazu Luque y al Dr. Jorge Alejandro Sanchez Garces por su guía y ayuda durante esta investigación, también agradezco a la Universidad Peruana Unión – Filial Juliaca que con sus docentes logran formar profesionales de éxito.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

SÍMBOLOS USADOS

- UPeU: Universidad Peruana Unión
- DTI: Dirección de Tecnologías de Información
- IDS: Sistema de detección de intrusos
- IPS: Sistema de prevención de intrusos
- GUI: Interfaz gráfica de usuario
- ISO: Organización internacional de normalización
- PC computadora personal
- IP internet protocol
- NIDS: Sistema de detección de intrusos de red
- HIDS: Sistema de detección de intrusos de host
- TCP: Protocolo de control de transmisión
- UDP: Protocolo de datagramas de usuario
- ICMP: Protocolo de control de mensajes de Internet
- DNS: Sistema de nombres de dominio
- HTTP: Protocolo de transferencia de hipertexto
- MiTM: Ataque de intermediario
- DDoS: Ataque de denegación de servicio
- VPN: Red privada virtual
- SSL: Capa de conexión segura
- MSG: Mensaje
- OSSIM: Gestión de información de seguridad de código abierto
- GPL: Licencia publica general
- SGSI: Sistema de gestión de la seguridad de la información
- S.O: Sistema operativo
- DAQ: Adquisición de datos
- FTP: Protocolo de transferencia de archivos

RESUMEN

Este proyecto se realiza con el propósito de poder brindar una herramienta adicional de detección y prevención de intrusos con el cual puedan monitorear el tráfico de la red y de esa manera evitar problemas de ataques o accesos a usuarios con permisos no autorizados. Snort es un sistema de detección de intrusiones (IDS) que permite el monitoreo del tráfico de red y la generación de alertas cuando los paquetes obtenidos indican que hay comportamientos sospechosos. Iniciamos esta investigación con la revisión de los componentes de Snort y todas las funcionalidades que nos puede brindar, luego procedemos a crear máquinas virtuales para que nos pueda ayudar a simular el monitoreo del tráfico así como algunos ataques, continuamos configurando snort en modo IDS/IPS, luego pasamos a crear reglas que nos ayuden a detectar los ataques y monitorizar el tráfico que se realizan y finalmente comprobamos la eficacia de su detección y prevención, para poder tener un buen control de la red usaremos la norma ISO 27001 el cual nos ayuda a conocer los métodos de evaluación del trafico mediante las etapas que esta nos brinda, se inicia con la fase de planificación donde analizaremos y diseñaremos las reglas según el trafico de la institución, pasamos a la fase de implementación donde elegimos a snort como IDS/IPS, configuramos el motor de amenazas así como nuestras reglas, continuamos con la fase de medir para evaluar que snort cumple con las necesidades que se espera y terminamos con la fase de mejorar haciendo algunas pruebas. En los resultados más relevantes tenemos las pruebas con ataques de denegación de servicio hacia la red configurada, snort cumple con su trabajo al detectar y bloquear las múltiples peticiones hechas por el usuario, así como el bloqueo del ingreso de los usuarios hacia nuestro servidor. Concluimos que snort es una muy buena herramienta opensource en el ámbito de IDS/IPS es posible implementarlo no solo en la UPeU si no en empresas que tengan deficiencias en el área de seguridad de la información.

Palabras clave: Detección de intrusos, Prevención de intrusos, Snort, ISO 27001.

ABSTRACT

This project is carried out with the purpose of being able to provide an additional tool of detection and prevention of intruders with which they can monitor the traffic of the network and in this way avoid problems of attacks or access to users with unauthorized permissions. Snort is an intrusion detection system (IDS) that allows the monitoring of network traffic and the generation of alerts when the packages obtained indicate suspicious behavior. We initiate this investigation with the revision of the components of Snort and all the functionalities that can offer us, then we proceed to create virtual machines so that it can help us to simulate the monitoring of the traffic as well as some attacks, we continue configuring snort in IDS / IPS mode , then we go on to create rules that help us detect the attacks and monitor the traffic that is carried out and finally we verify the effectiveness of their detection and prevention, in order to have a good control of the network we will use the ISO 27001 standard which helps us know the methods of evaluation of the traffic through the stages that this provides, begins with the planning phase where we analyze and design the rules according to the traffic of the institution, we move to the implementation phase where we choose snort as IDS / IPS, we configure the threat engine as well as our rules, we continue with the measurement phase to evaluate which snort meets the necessary we expect and finish with the improvement phase doing some tests. In the most relevant results we have the tests with denial of service attacks against the configured network, snort accomplishes its work by detecting and blocking the multiple requests made by the user, as well as blocking the entry of users to our server. We conclude that snort is a very good opensource tool in the scope of IDS / IPS, it is possible to implement it not only in UPeU but also in companies that have deficiencies in the area of information security.

Keywords: Intruder detection, Intruder prevention, Snort, ISO 27001.

CAPÍTULO I. El problema

1.1 Identificación del problema

Según el informe del Banco Inter Americano de Desarrollo (BID) que se encarga del financiamiento de proyectos viables económico, social e institucional en la región de américa latina y la OEA anuncia que Perú hoy en día aún no cuenta con una estrategia nacional de ciberseguridad, sin embargo, si tiene lo necesario para combatir las amenazas del ciberespacio. Viendo todos estos detalles llegamos a la conclusión de que cualquier empresa ya sea una institución educativa es importante que pueda contar con una buena seguridad de su información (PortalAndina, 2017, p. 3).

Según el estudio realizado por el diario la republica muestra que Perú es el quinto país con más ataques cibernéticos en Latinoamérica este estudio también revela que en el sector financiero sufre más de 6.6 millones de ataques por día, con un 11.22% Perú ha llegado a ser el quinto país de Latinoamérica que recibe más ataques cibernéticos (LaRepública, 2015, p. 5).

El aumento del tráfico de datos en la red, y algunos ataques informáticos en la Universidad Peruana Unión – Filial Juliaca, actualmente genera bastantes problemas como la lentitud de algunos servicios informáticos, también como la caída de algunos sistemas informáticos de la UpeU – Juliaca. Actualmente la universidad cuenta con un firewall basado en hardware que se encarga de eliminar virus gusanos y otras amenazas llamado Fortigate, sin embargo, este firewall cumple con bastantes funciones y no específicamente en el monitoreo del tráfico de la red, Según los datos estadísticos del Fortigate el cual se encarga de la gestión unificada de amenazas, Anexo C - H. Nos dice que las principales amenazas en la red son las visitas a sitios web maliciosos y al control de cortafuegos Firewall. Nos muestra que en un rango del 1% al 100% siendo el 100% el puntaje de tráfico más alto por día. También nos muestra que un rango critico de ataques, tenemos los ataques en inyecciones por comandos a los dispositivos Routers, también nos muestra el virus Adware / AirPush! Android que es un virus de aplicativo móvil que contiene un componente de publicidad de terceros que muestra contenido publicitario en el panel de notificaciones del

dispositivo este aplicativo que se instala silenciosamente también puede reunir información y reenviar detalles del dispositivo. También tenemos el conjunto de red de robots llamado botnet el cual se encarga de controlar nuestros ordenadores o servidores de forma remota pudiendo de esa manera realizar ataques de denegación de servicio DDoS.

Es por eso que es necesario minimizar los ataques cibernéticos y monitorizar el tráfico de datos ya que provocan una lentitud en la red y caídas de los sistemas de información.

1.2 Justificación

En el presente proyecto se pretende diseñar e implementar un sistema de prevención y detección de intrusos en la red de la Universidad Peruana Unión – Juliaca, el cual ayudara a monitorizar el tráfico entrante y saliente, pudiendo así prevenir que un usuario no autorizado pueda obtener algún tipo de beneficio ya sea directamente económico pidiendo un rescate por la información, los llamados Ramsonware o simplemente con la finalidad de poder espiar a la institución educativa con un acceso no autorizado. Se realizarán pruebas en interno para de esa manera verificar la eficacia de nuestro IDS/IPS, y usando la metodología ISO 27001 el cual nos ayudara en ver que se puede mejorar y en que es lo que se debe de estar alertas.

Para poder contrarrestar este tipo de problemas es que se optó por usar un (IDS/IPS) ya que nos ayuda a tener un mejor control y una buena seguridad en nuestras redes cibernéticas, ambos sistemas de (IDS/IPS) nos ayudaran a monitorear de manera más eficiente los servicios que se tienen (Martinez, Bermúdez, & Ocampo, 2017).

Al pasar los años la Universidad Peruana Unión no solo fue creciendo estructuralmente si no educativamente es por ello que el personal administrativo aumento y por ende los equipos informáticos en cada área de trabajo, esto hace que cada área pueda contar con información confidencial, de tal forma que tiene que requerir de una buena seguridad de su información, si bien el uso del módulo de IPS del Fortinet es muy eficiente, pero a la vez muy costoso es por eso que se optó por implementar un sistema de detección y prevención de intrusos.

Snort es un Sistema de Detección y prevención de Intrusos (IDS/IPS) open source que también cuenta con un lenguaje de reglas configurables, una de las ventajas de usar Snort es que también tiene la capacidad de poder usarse como un sniffer con el cual podremos ver en tiempo real lo que sucede en nuestra red, todo el tráfico que se genera, también nos permite guardar en un archivo log el trafico escaneado para luego poder hacer un análisis fuera de la red Roesch (2010).

1.3 Presunción filosófica

A lo largo de nuestra formación académica adquirimos diversas aptitudes en los 5 años en las aulas universitarias donde a la vez también adquirimos diferentes capacidades y competencias en el desenvolvimiento de nuestra carrera de Ingeniería de Sistemas. La formación que recibimos en nuestra institución universitaria no solo está basada en conocimientos académicos, sino que también recibimos una formación con principios y valores cristianos y en esta investigación que desarrollamos un modelo de un sistema de detección y prevención de intrusos, mostramos las habilidades académicas que adquirimos a lo largo de nuestra formación universitaria.

Proverbios 27:12 dice: "El prudente ve el peligro y lo evita; el inexperto sigue adelante y sufre las consecuencias."

Esta investigación se basa en nuestros principios y valores cristianos con la voluntad de desarrollar e implementar un buen producto en el área que se ha propuesto.

1.4 Objetivos

- **Objetivo general.**

  Implementar un sistema de detección y prevención de intrusos (IDS/IPS), basado en la norma ISO 27001, para el monitoreo perimetral de la seguridad informática, en la red de la Universidad Peruana Unión – Filial Juliaca

- **Objetivos específicos.**
  - ✓ Analizar y diseñar las reglas según el tráfico y los ataques más relevantes en la comunidad.
  - ✓ Instalar y configurar el IDS/IPS Snort

- ✓ Monitorear y análisar del tráfico de datos, seguridad y disponibilidad en la red.
- ✓ Pruebas del análisis del tráfico de la red y bloqueo de tráfico no permitido.

CAPÍTULO II. Revisión de la literatura

2.1 Antecedentes de la investigación.

En la tesis de licenciatura en ciencias de la computación realizado por Lozano (2017) con el título de estudio, SnorUNI: Aplicación móvil para Sistemas de Detección y Prevención de Intrusiones basados en Snort. Desarrolló un aplicativo móvil con el que pueda permitir administrar la seguridad y de esa manera obtener información sobre las alertas de una manera más rápida. Durante el desarrollo de la aplicación móvil se utilizó la metodología SCRUM. En esta sección se estudiarán sus principales características y fases de implementación. Un trabajo más enfocado en Snort, fue el análisis de rendimiento para detección utilizando la data set IDEVAL, los resultados muestran la gran efectividad de Snort para detectar intrusiones si es que tiene su base de firmas correctamente actualizada.

En la tesis de licenciatura en ingeniería de sistemas realizado por Alegria (2016) con el título de estudio, Implementación de un sistema de seguridad (ids/ips) open source basado en raspberry para la red del ministerio público sede puno. Implementó un sistema de seguridad con el cual pueda permitir reforzar y alertar ante posibles intrusiones no deseadas de la red, por medio de un dispositivo de bajo costo que es Rasperry. La propuesta metodológica que propone es la implementación de una nueva tecnología de planeamiento y ejecución de la defensa en niveles de seguridad y cambiar la política actual al más alto nivel, dejando de lado el defensivo concepto medieval de murallas. Como resultado se logró implementar un sistema de seguridad que nos permite restringir accesos no deseados para mantener la confidencialidad, disponibilidad e integridad de la información del Ministerio Público sede Puno.

En la tesis de licenciatura en ingeniería de sistemas realizado por Vieira & Ramos 2018) con el título de estudio. Implementación de una solución de seguridad perimetral Open Source en La Red Telemática de la Universidad Nacional Pedro Ruiz Gallo. Se implementó una solución de seguridad perimétrica open source para que cubra los requerimientos de una red perimetral DMZ con el software pfSense. Como marco metodológico puso en práctica solucionar los problemas de seguridad perimetral haciendo uso de las normas ISO 27001

dimensionadas a las exigencias de las normativas de la red telemática. En los resultados se determinó que el promedio de ataques de accesos no autorizados exitosos antes de la implementación del sistema que fue de 104.5 con desviación estándar de 98.33 ingresos, sin embargo después de la implementación el promedio fue de tan solo del 4.88 con una desviación estándar de 3.18 ingresos, se puede observar existe una diferencia del 99.63 de ataques de accesos no autorizados exitosos en promedio a favor del firewall de seguridad perimetral, también se puede observar que el número máximo y mínimo de ingresos antes de la implementación del firewall fue de 150 y 10 respectivamente, mientras que después de la implementación fue de 10 y 1 respectivamente.

En la tesis de maestría en conectividad y redes realizado por Diaz (2018) con el título de estudio. Propuesta metodológica y simulación de la implementación de un siem basado en la norma iso 27001 y/o 27002. Desarrolló una Metodología de implementación del SIEM, para cubrir su ausencia a nivel global, capaz de ser utilizada en modelos de negocios de TI (Tecnología de la información), como herramienta práctica, eficaz y eficiente que facilite la compleja tarea de implementación de la tecnología mencionada a profesionales de la rama, interesados en proteger los activos de información de la organización. En la metodología propuesta opto por usar los métodos que nos brindan la norma ISO 27001 e ISO 27002. Los resultados obtenidos por cada uno de sus subprocesos y sus actividades respectivamente, donde la mayoría de ellos fueron éxitos y otros destacan algunos problemas que deben ser tomados en cuenta para su corrección.

En la tesis de licenciatura en ingeniería de sistemas e informática realizado por Troya (2015) con el título de estudio. Sistema de detección de intrusos en redes mediante el método heurístico para el gobierno municipal de la ciudad de otavalo. Desarrolló un método heurístico para resolver un problema de optimización mediante una aproximación intuitiva, en la que se usa de manera inteligente para obtener una buena solución. Se utiliza evidencias del análisis de los procesos internos que se realizarán en la detección de intrusos. Para la metodología de desarrollo de software aplicada es la XP, donde brinda una programación organizada, menos cantidad de errores, satisfacción en la programación, solución de errores de programas, versiones nuevas. Como resultado se obtuvo las búsquedas inteligentes de intrusos en la red por el método heurístico la cual se podrá estar informado puntualmente de la actividad en la red local a la que está conectado nuestro PC, a esta funcionalidad

informativa se une la ventaja de saber si alguna persona desapercibida está conectada a la red.

2.2 Marco teórico.

2.2.1 Seguridad Informática

"Podemos definir que la seguridad informática es el proceso de prevenir y detectar el uso no autorizado de un sistema informático. Implica el proceso de protección contra intrusos acerca del uso de nuestros recursos informáticos con intenciones maliciosas". (Universidad Internacional de Valencia, 2016, p. 4). A medida que muchas personas se conectan a Internet eso conlleva a que continúan aumentando a un ritmo rápido, hasta el punto que cada vez más de nosotros estamos haciendo nuestras propias redes de computadoras en el hogar. Con estos, podemos tener las ventajas de tener un gran ancho de banda, acceso instantáneo a Internet y hacer que esta conexión sea accesible para muchas PC dentro y fuera de la casa. Pero para aquellos que no conocen la seguridad de la computadora, desconocen por completo los riesgos que pueden estar exponiendo a su computadora también. Sin utilizar una solución de seguridad informática adecuada, su computadora puede ser pirateada o infectada con virus, adware o spyware.

Estas formas de malware que pueden desempeñar una función de archivos dañinos al convertir una computadora inutilizable, destruir datos importantes que almacena, otorgar el control completo de una PC a otra persona. Permitir que alguien robe los datos en su computadora, registrar sus pulsaciones de teclado y otorgarle acceso a un tercero a su cuenta bancaria en línea, permitir que alguien use su computadora para atacar una computadora que pertenece a otra persona.

2.2.2 Áreas que están implicadas en la seguridad informática.

- Confidencialidad: Solamente usuarios autorizados pueden acceder a nuestros sistemas y datos e información.
- Integridad: Solamente usuarios autorizados deben ser capaces de modificar los datos cuando sea necesario.
- Disponibilidad: Los datos e información deben estar disponibles para los usuarios cuando sea necesario.

- Autenticación: Estás seguro que realmente estas comunicándote con los que piensas que te estás comunicando.

2.2.3 IDS/IPS.

2.2.3.1 IDS.

Un sistema de detección de intrusos (IDS) es un sistema que monitorea el tráfico de la red en exploración de actividad sospechosa y emite alertas cuando se descubre alguna acción. Si bien la detección de anomalías y los informes son la misión fundamental, algunos sistemas de detección de intrusos son capaces de tomar medidas cuando se detecta alguna acción maliciosa o tráfico extraño, incluido el bloqueo del tráfico enviado desde direcciones IP sospechosas. Sin embargo los sistemas de detección de intrusos monitorean las redes en busca de alguna acción maliciosa, del mismo modo son propensos a falsas alarmas (falsos positivos). Como resultado, las organizaciones necesitan adecuar sus productos IDS cuando los instalan por primera vez. Eso significa configurar adecuadamente sus sistemas de detección de intrusos para observar cómo se ve el tráfico normal en su red en comparación con la acción potencialmente maliciosa (SeguridadGJSA, 2015, p. 4).

2.2.3.2 Tipos de IDS

"El término IDS (Sistema de detección de intrusiones) hace referencia a un mecanismo que, sigilosamente, escucha el tráfico en la red para detectar actividades anormales o sospechosas, y de este modo, reducir el riesgo de intrusión" (SeguridadGJSA, 2015, p. 4).

Existen dos claras familias importantes de IDS:

El grupo N-IDS (Sistema de detección de intrusiones de red), que garantiza la seguridad dentro de la red.

El grupo H-IDS (Sistema de detección de intrusiones en el host), que garantiza la seguridad en el host.

Un N-IDS necesita un hardware especial. Éste forma un sistema que puede comprobar paquetes de información que viajan por una o más líneas de la red para revelar si se ha producido alguna acción maliciosa o inusual. El N-IDS pone uno o más de los

adaptadores de red exclusivos del sistema en modo promiscuo. Éste es una especie de modo intangible en el que no tienen dirección IP. Ni tienen una serie de protocolos asignados. Es común hallar diversos IDS en diferentes partes de la red. Por lo frecuente, se colocan sistemas de detección fuera de la red para observar los posibles ataques, así como también se colocan sistemas internos para examinar solicitudes que hayan pasado a través del firewall o que se han realizado internamente (SeguridadGJSA, 2015, p. 4).

El H-IDS se encuentra en un host individual. Por lo tanto, su software cubre una amplia variedad de sistemas operativos como Windows, Solaris, Linux, HP-UX, Aix etc. El H-IDS actúa como un demonio o servicio estándar en el sistema de un host. Tradicionalmente, el H-IDS analiza la información personal almacenada en registros como registros de sistema, mensajes, lastlogs y wtmp y además captura paquetes de la red que se introducen/salen del host para poder comprobar las señales de intrusión como ataques por denegación de servicio, puertas traseras, troyanos, intentos de acceso no autorizado, realización de códigos malignos o ataques de inundación de búfer (SeguridadGJSA, 2015, p. 4).

2.2.3.3 Funcionamiento IDS

Los principales métodos que usa un N-IDS para informar y bloquear intrusiones son:

"Reconfiguración de dispositivos externos firewalls o ACL en routers: Comando enviado por el N-IDS a un dispositivo externo como un filtro de paquetes o un firewall para que se reconfigure inmediatamente y así poder bloquear una intrusión. Esta reconfiguración es posible a través del envío de datos que expliquen la alerta en el encabezado del paquete" (SeguridadGJSA, 2015, p. 5).

"Envío de una trampa SNMP a un hipervisor externo: Envío de una alerta (y detalles de los datos involucrados) en forma de un datagrama SNMP a una consola externa como HP Open View Tivoli, Cabletron, Spectrum, etc" (SeguridadGJSA, 2015, .p 5).

"Envío de un correo electrónico a uno o más usuarios: Envío de un correo electrónico a uno o más buzones de correo para informar sobre una intrusión seria" (SeguridadGJSA, 2015, p. 5).

"Registro del ataque: Se guardan los detalles de la alerta en una base de datos central, incluyendo información como el registro de fecha, la dirección IP del intruso, la dirección IP del destino, el protocolo utilizado y la carga útil" (SeguridadGJSA, 2015, p. 5).

"Almacenamiento de paquetes sospechosos: Se guardan todos los paquetes originales capturados y/o los paquetes que dispararon la alerta" (SeguridadGJSA, 2015, p. 5).

"Apertura de una aplicación: Se lanza un programa externo que realice una acción específica (envío de un mensaje de texto SMS o la emisión de una alarma sonora)" (SeguridadGJSA, 2015, p. 5).

"Envío de un 'ResetKill': Se construye un paquete de alerta TCP para forzar la finalización de una conexión (sólo válido para técnicas de intrusión que utilizan el protocolo de transporte TCP)" (SeguridadGJSA, 2015, p. 5).

"Notificación visual de una alerta: Se muestra una alerta en una o más de las consolas de administración" (SeguridadGJSA, 2015, p. 5).

2.2.3.4 IPS

Un Sistema de prevención de intrusiones (IPS) es una tecnología de seguridad de red prevención de amenazas que examina los flujos de tráfico de la red para detectar y prevenir las vulnerabilidades de vulnerabilidad. Las vulnerabilidades de vulnerabilidad por lo general vienen en forma de entradas maliciosas a una aplicación o servicio de destino que los atacantes usan para interrumpir y obtener el control de una aplicación o máquina. Tras una explotación exitosa, el atacante puede deshabilitar la aplicación de destino (lo que resulta en un estado de denegación de servicio), o puede acceder a todos los derechos y permisos disponibles para la aplicación comprometida. "Siendo esta última una característica que distingue a este tipo de dispositivos de los llamados Sistemas de Detección de Intrusos o Intrusion Detection Systems 'IDS' en sus siglas en inglés." (PandaSecurity, 2018, p. 4)

Entre las diferentes capacidades (como es el IDS), el administrador debe estar alarmado por el reconocimiento de interrupciones o movimientos vengativos, mientras que es selectivo a un Sistema de Prevención de Intrusos (IPS) para desarrollar enfoques de seguridad para garantizar el hardware o el sistema de un asalto (PandaSecurity, 2018, p. 4)

Otras funciones de estos dispositivos de red, son las de grabar información histórica de esta actividad y generar reportes para dar un mejor seguimiento.

Según SeguridadGJSA (2015, p. 4), los IPS se clasifican en cuatro diferentes tipos:

- Basados en Red Lan (NIPS): monitorizan la red lan en busca de tráfico de red sospechoso al analizar la actividad por protocolo de comunicación lan.
- Basados en Red Wireless (WIPS): monitorean la red inalámbrica en busca de tráfico sospechoso al analizar la actividad por protocolo de comunicación inalámbrico.
- Análisis de comportamiento de red (NBA): Examina el tráfico de red para identificar amenazas que generan tráfico inusual, como ataques de denegación de servicio ciertas formas de malware y violaciones a políticas de red.
- Basados en Host (HIPS): Se efectúa mediante la instalación de paquetes de software que monitoriza un host único en busca de actividad sospechosa.

Los IPS categorizan la forma en que detectan el tráfico malicioso:

- Detección basada en firmas: como lo hace un antivirus.
- Detección basada en políticas: el IPS requiere que se declaren muy específicamente las políticas de seguridad.
- Detección basada en anomalías: en función con el patrón de comportamiento normal de tráfico.

2.2.3.5 Características de un IPS.

El IPS puede cuadrar las interrupciones, prestar poca atención a la convención del tráfico y reconfigurar un dispositivo externo. Esto implica que el IPS puede canalizar y cuadrar parcelas en modo local, utiliza estrategias, por ejemplo, eliminando una asociación, eliminando paquetes de restricción, obstaculizando un interloper, etc. (SeguridadGJSA, 2015, p. 5).

2.2.4 Herramientas IDS/IPS Open Source

2.2.4.1 Snort

"Todo un veterano cuando se trata de análisis de paquetes. La primera versión vio la luz allá por 1998. Cabe mencionar que en aquel momento no se contempló como IDS puro, pero fue evolucionando hasta ese punto poco a poco." (Bricata, 2016, p. 3).

Snort es un sistema de detección de intrusos de código abierto (IDS) y un sistema de protección contra intrusos (IPS) desarrollado originalmente en 1998. Snort hizo que utilizar la nueva inteligencia de amenazas fuera increíblemente sencillo para escribir Reglas de Snort que detectarían amenazas emergentes.

El sitio web de Snort señala: A diferencia de las firmas, las reglas se basan en la detección de la vulnerabilidad real, no en un exploit o en una pieza de datos única. Desarrollar una regla requiere una comprensión aguda de cómo funciona realmente la vulnerabilidad.

Para decirlo de otra manera, Snort Rules proporciona una definición simple que ayuda a especificar características únicas del tráfico de red, activa una alerta cuando se cumplen esas condiciones y elimina o bloquea la comunicación según lo desee la configuración del usuario.

Una de las principales desventajas de Snort es que no cuenta con una interfaz gráfica administrable, por ende, no es fácil de administrar. Pero, existen herramientas opensource que pueden compensar eso, como Snorby o Squil.

2.2.4.2 Suricata

"Podríamos decir que la única razón para no utilizar Snort es estar utilizando Suricata. Aunque se trata de una herramienta IDS de arquitectura distinta, se comporta de la misma manera que Snort y usa las mismas firmas." (Márquez, 2017, p. 3).

Ahora, veremos algunos de los puntos clave de este sistema de detección de intrusiones avanzado:

- “Multi-hilo: Snort se ejecuta en modo uni-hilo y por tanto solo puede aprovechar un núcleo de la CPU al mismo tiempo. Suricata aprovecha los procesadores multi-núcleo y multi-threading. Existen benchmarks que demuestran una considerable diferencia de rendimiento” (Márquez, 2017, p. 4).
- “Aceleración mediante hardware: es posible utilizar tarjetas gráficas para inspeccionar tráfico de red” (Márquez, 2017, p. 4).
- “Extracción de ficheros: si alguien se descarga malware en nuestro entorno, es posible capturarlo y estudiarlo desde Suricata” (Márquez, 2017, p. 4).
- “LuaJIT: nos proporcionará el poder que nos falta mediante scripting, pudiendo combinar varias reglas, para buscar elementos con mayor eficiencia” (Márquez, 2017, p. 4).
- “Más que paquetes: Suricata no solo es capaz de analizar paquetes, también puede revisar los certificados TLS/SSL, peticiones DNS, solicitudes HTT” (Márquez, 2017, p. 4).

Al igual que Snort, Suricata depende de los estándares y, a pesar de que ofrece similitud con las reglas de Snort, también presentó diferentes cadenas, que se compara con el límite hipotético de las pautas más en los sistemas más rápidos, en los volúmenes de tráfico. Más enorme, en un equipo similar, (Márquez, 2017, p. 4).

2.2.4.3 Bro

Bro es un sistema de detección de intrusos que funciona de manera diferente a otros sistemas debido a su enfoque en el análisis de redes. Mientras que los motores basados en reglas están diseñados para detectar una excepción, Bro busca amenazas específicas y dispara alertas.

Si bien Bro puede ser utilizado como un IDS tradicional, los usuarios usan Bro con más frecuencia para registrar el comportamiento detallado de la red. Por ejemplo, se puede usar para mantener registros a largo plazo de todas las solicitudes y resultados HTTP, o tablas que correlacionan direcciones MAC e IP.

“El verdadero potencial de Bro es el Intérprete de Políticas Script. Con su propio lenguaje de administración (Bro-Script) que nos ofrece varias posibilidades.” (Márquez, 2017, p. 3).

2.2.4.4 Kismet (Wireless)

De manera similar, cuando Snort se convirtió en el estándar para el examen de interrupción en rojo, Kismet se convirtió en una referencia para IDS remotos. Un IDS Wireless tiene menos que ver con la carga de paquetes, y más con las ocasiones que ocurren en el sistema (Márquez, 2017, p. 3).

WIDS encontrará, por ejemplo, Puntos de entrada falsos o simulados (Rogue AP) que inclusive podrían ser creados sin malicia por un empleado de la compańía, abriendo una grieta en la red. En cualquier caso, si nuestra red dispone de repetidores y puntos de entrada WiFi, es espléndido para impedir intentos de substitución de nuestra SSID/Mac y por consiguiente, impedir ataques MiTM (Márquez, 2017, p. 3).

2.2.5 Herramientas IDS/IPS comerciales.

2.2.5.1 McAfee Network Security Platform (NSP)

McAfee Network Security Platform es un sistema de prevención de intrusiones (IPS) de próxima posteridad que redefine el carácter en que las organizaciones bloquean las amenazas. Bloquea los ataques nuevos y desconocidos con un examen basada en firmas y sin firmas con soporte para VMware NSX y OpenStack permite a las organizaciones agrupar la seguridad a través de redes físicas y virtuales (McAfee, 2018, p .5).

2.2.5.2 Radware DefensePro IPS

DefensePro DDoS Protection IPS y realiza modelos de seguridad, Radware DefensePro es la mejor garantía contra DDoS en el mercado. DefensePro es el arreglo ganador de honor de Radware que brinda la mejor garantía de DDoS a las crecientes agresiones del sistema, a pesar de las agresiones conocidas basadas en la vulnerabilidad (Radware, 2018, p. 5).

2.2.5.3 StoneSoft (McAfee)

Su portafolio de productos incluye dispositivos de firewall / VPN, IPS (sistemas de prevención y detección de intrusos) y sistemas VPN SSL, cada uno apto como dispositivos de hardware, software y dispositivos virtuales certificados por VMware. Cada uno de los componentes, así como los dispositivos de terceros, se pueden manejar desde el centro de

Gestión de Stonesoft. La carpeta de productos se diferencia a través de tecnologías únicas de concurrencia y armonía de carga basadas en la tecnología más antigua StoneBeat de la compañía, desarrollada originalmente para Check Point FireWall-1. (Forcepoint, 2018, p. 5).

2.2.5.4 IBM Security Network Intrusion Prevention System

Los dispositivos y sistemas del Sistema de prevención de intrusiones (IPS) de IBM Security Network bloquean automáticamente los ataques maliciosos al término que preservan el ancho de banda y la disponibilidad. Se puede aprovechar la Interfaz de administración local de IPS, la interfaz de administración local basada en la web, para efectuar actualizaciones, efectuar ajustes y ampliar los ajustes de configuración según sea elemental. Se puede utilizar SiteProtector para modificar las políticas si está administrando múltiples productos de prevención o detección de intrusos (IBM, 2018, p. 6).

2.2.5.5 Cisco Sourcefire IPS

Cisco Sourcefire recibe nuevas reglas de política y firmas cada dos horas, por lo que su seguridad está siempre actualizada. Cisco Talos aprovecha la red de detección de amenazas más inmensa del mundo para ofrecer efectividad de seguridad a cada producto de seguridad de Cisco. Este alcance de amenazas líder en la competencia funciona como un sistema de alerta temprana que se actualiza frecuentemente para detectar nuevas amenazas (Cisco, 2018, p. 7).

2.2.6 Seguridad perimetral

La seguridad perimetral informática no deja de sostener el mismo valor que la general. De hecho, también son todos los sistemas destinados a proteger de intrusos tu ámbito. La única variedad es que, en lugar de un espacio físico, se protegen las redes privadas de tu sistema informático. Se se refiere de un perfil de defensa, similar a las alarmas de una oficina. La seguridad total no existe ni en el mundo físico ni en el informático, pero reduce muchísimo el riesgo a que nos roben nuestros datos o, inclusive, que puedan desvanecerse. Las funciones de una buena seguridad perimetral son: (accensit, 2016, p. 6).

La seguridad perimetral que protege tus redes debe cumplir cuatro funciones básicas:

- Resistir a los ataques externos.
- Identificar los ataques sufridos y alertar de ellos.
- Aislar y segmentar los distintos servicios y sistemas en función de su exposición a ataques.
- Filtrar y bloquear el tráfico, permitiendo únicamente aquel que sea absolutamente necesario (accensit, 2016).

“Las herramientas que hay en la seguridad perimetral son los cortafuegos, los sistemas detección y prevención de intrusos, honeypots, pasarelas, antivirus y antispam.” (accensit, 2016, p. 4).

2.2.7 SNORT Sistema de detección y prevención de intrusos.

Snort es un Sistema de Detección de Intrusos (IDS) basado en red (IDSN) open source. Cuenta con un lenguaje de elaboración de reglas en el que se pueden establecer los estándares que se utilizarán al momento de monitorizar el sistema. Conjuntamente, ofrece una serie de reglas y filtros ya predefinidos que se pueden adecuar durante su instalación y configuración para que se pueda adaptar a lo que deseamos (Ortego Delgado, 2017, p. 7).

Una de las ventajas de este sistema es que puede trabajar como sniffer podemos visualizar en consola y en tiempo real qué ocurre en nuestra red, todo nuestro tráfico, lista de paquetes, permite almacenar en un archivo los logs para su posterior análisis, un examen offline o como un IDS estándar en este caso NIDS. Cuando un paquete coincide con algún patrón establecido en las reglas de configuración, así se sabe cuándo, de dónde y cómo se produjo el ataque (Ortego Delgado, 2017, p. 8).

Aspectos que destacan en Snort:

- **Configuración:** Se usa LuaJIT para la configuración. La sintaxis de configuración es simple, consistente y ejecutable. Los complementos de LuaJIT para opciones de reglas y registradores también son compatibles.
- **Detección:** fue trabajado en estrecha colaboración con Cisco Talos para actualizar las reglas, para de esa manera satisfacer las necesidades que demandan, incluida una característica que denominan "buffers pegajosos". Con el uso del motor de búsqueda

Hyperscan, los patrones rápidos de expresiones regulares hacen que las reglas sean más rápidas y más precisas.

- **HTTP:** Se tiene un inspector de HTTP nuevo y con estado que actualmente maneja el 99 por ciento de los casos de HTTP Evader, y pronto se cubrirán todos. También hay muchas características nuevas, incluyendo nuevas opciones de reglas. El soporte HTTP / 2 que está en desarrollo.
- **Rendimiento:** Se mejoró sustancialmente el rendimiento de la inspección profunda de paquetes. Snort 3 es compatible con múltiples subprocesos de procesamiento de paquetes, y se escala de forma lineal con una cantidad de memoria mucho menor requerida para configuraciones compartidas, como los motores de reglas.
- **Registro de eventos JSON:** se pueden utilizar para integrarse con herramientas como Elastic Stack.
- **Complementos:** Snort 3 fue diseñado para ser extensible y hay más de 225 complementos de varios tipos. Es fácil agregar su propio códec, inspector, acción de regla, opción de regla o registrador. Las reglas de SO son complementos también, y es mucho más fácil agregar los que queremos usar.

Así como nos muestra en la investigación de Alegria (2016) el cuadro comparativo de antes y después de la implementación del sistema propuesto, vemos una mejora notoria en el monitoreo de los datos en la red, de igual manera el costo que fue propuesto para esa investigación fue un costo accesible para cualquier empresa al ser Snort una herramienta Open Source. Veremos una breve comparación de algunos IDS OpenSource según la investigación de McCranie (2011).

Tabla 1.

Comparación de snort con otros IDS open source

Características	Bro	Snort	Suricata
Multi-hilo	No	Si	Si
Soporte para IPv6	Si	Si	Si
Ip reputación	Algo	Si	Si
Detección automática de protocolos	Si	Si	Si
Aceleración con GPU	No	Si	Si
Variables globales	Si	Si	Si
GeoIP	Si	Si	Si
Análisis Avanzado de HTTP	Si	Si	Si
Login de acceso para HTTP	Si	Si	Si
Login de acceso para SMB	Si	Si	Si

Fuente: Alegria (2016).

2.2.7.1 Arquitectura de snort

Figura 1. Arquitectura del motor de detección de snort.
Fuente: SeguridadYredes, 2019, p. 6.

Los preprocesadores son pequeños plugins programados normalmente en C que sirven para revisar los paquetes provenientes del decodificador. El método que realiza sobre los paquetes es para darle forma de modo que se pueda explicar la información de los paquetes de forma más sencilla y lógica. Una vez reordenados los paquetes, al pasar por el motor de Detección se le aplicarán las Reglas en inspección de patrones de ataques, virus, información, etc. Los preprocesadores pueden desfragmentar paquetes, ordenarlos, decodificar URLs. Estos preprocesadores se configuran en el registro de configuración etc/snort.conf. (SeguridadYredes, 2019, p. 6).

2.2.7.2 Snort IDS

Snort IDS permite monitorear todos los paquetes que atraviesan la red en la cual se ha instalado. Estos paquetes son analizados y es posible establecer qué acciones se llevarán a cabo a partir de reglas. La postura de Snort se establece a partir de un registro de configuración que responde al nombre de snort.conf. En este registro se especifican las distintas opciones que delimitan cómo se comportará Snort y de qué manera trabajará. De este modo, para iniciar snort en modo promiscuo y empezar a ver los paquetes por consola, se debe ejecutar el siguiente comando: snort –v –o [interfaz]. Una vez que snort ya se está ejecutando, podrá visualizarse en la consola todas aquellas actividades que ocurran a nivel de red previa configuración hecha (Cotoira, 2014).

2.2.7.3 Snort IPS

Esta labor utiliza la solución de código abierto Snort para permitir IPS e IDS. Snort realiza las siguientes acciones:

- Sensor de snort: controla el tráfico para detectar anomalías según las políticas de seguridad configuradas (que, incluyendo las firmas, estadísticas, análisis de protocolo, etc.) y envía mensajes de alerta a Alert / Reporting servidor. El sensor Snort se implementa como un servicio de contenedor virtual en el enrutador.
- Almacén de firmas: aloja los paquetes de firmas de Cisco que se actualizan periódicamente. Los paquetes se descargan a los sensores ya sea periódicamente o a pedido. Paquetes de firmas validados se publican en Cisco.com. Según la configuración, los paquetes de firmas se pueden descargar desde Cisco.com.

Internamente del archivo de configuración de Snort es elemental modificar algunos parámetros para que el sistema pueda ejecutarse como un IPS INLINE. Indagar los valores y cambiarlos en el sitio adecuado de snort.conf (Cisco, 2015).

2.2.7.4 Reglas Snort

Una de las funcionalidades más completas y poderosas de Snort, es la posibilidad de establecer reglas. En ciertos casos, es primordial configurar determinadas alertas para determinados tipos de paquetes. Es frecuente, que, frente a una vulnerabilidad explotable remotamente en algún equipo crítico, será esencial solventar la situación mediante la configuración de alguna regla hasta que la vulnerabilidad tenga alguna solución (Cotoira, 2014).

Las reglas o firmas son los modelos que se buscan internamente en los paquetes de datos. Las reglas de Snort son utilizadas por el motor de detección para examinar los paquetes recibidos y generar las alertas en caso de existir similitud entre el contenido de los paquetes y las firmas. El archivo snort.conf permite añadir o eliminar clases enteras de reglas. Se pueden desactivar toda una categoría de reglas comentando la línea de la misma o bien algunas que veamos que no son necesarias (Roesch, 2012).

2.2.7.5 Categoría de reglas

Hay cuatro categorías de reglas para estimar un paquete. Donde se especifica dicha regla especifica o una establecida por Snort. Estas cuatro categorías están divididas a su vez en dos grupos, las que tienen contenido y las que no tienen contenido. Hay reglas de protocolo, reglas de contenido genéricas, reglas de paquetes mal formados y reglas IP.

- Reglas de Protocolo. Las reglas de protocolo son reglas las cuales son dependientes del protocolo que se está analizando, por ejemplo, en el protocolo Http está la palabra reservada uricontent.
- Reglas de Contenido Genéricas. Este tipo de reglas permite especificar patrones para buscar en el campo de datos del paquete, los patrones de búsqueda pueden ser binarios o en modo ASCII, esto es muy útil para buscar exploits los cuales suelen terminar en cadenas de tipo "/bin/sh".
- Reglas de Paquetes Malformados. Este tipo de reglas especifica características sobre los paquetes, concretamente sobre sus cabeceras las cuales indican que se está produciendo algún tipo de anomalía, este tipo de reglas no miran en el contenido ya que primero se comprueban las cabeceras en busca de incoherencias u otro tipo de anomalía.
- Reglas IP. Este tipo de reglas se aplican directamente sobre la capa IP, y son comprobadas para cada datagrama IP, si el datagrama luego es Tcp, Udp o Icmp se realizará un análisis del datagrama con su correspondiente capa de protocolo, este tipo de reglas analiza con contenido y sin él Roesch (2012, p .15).

2.2.7.6 Estructura de una regla

En una forma básica, una regla de Snort está compuesta en dos partes qu son encabezado y opciones

Figura 2. Estructura de una regla y su cabecera.
Fuente: Roesch (2012).

2.2.7.7 Cabecera de una regla

"La cabecera permite establecer el origen y destino de la comunicación, y sobre dicha información realizar una determinada acción. La cabecera contiene algunos criterios para unir la regla con un paquete y dictar qué acción debe tomar una regla. Su estructura es: <acción> <protocolo> <red origen> <puerto origen> <dirección> <red destino> <puerto destino>." (Roesch, 2012).

La estructura general de la cabecera una regla es la que vemos en la siguiente tabla.

Tabla 2.

Estructura de cabecera regla en snort.

Estructura de la cabecera de una regla snort						
Acción	Protocolo	Red origen	Puerto origen	Dirección	Red destino	Puerto destino
Alert	Tcp	$EXTERNARL_NET	Any	->	$HOME_NET	53

Fuente: Roesch (2012).

Veremos el significado de dicha estructura:

- Protocolo. Permite establecer el protocolo de comunicaciones que se va a utilizar. Los posibles valores son: TCP, UDP, IP e ICMP.
- Red de origen y red de destino. Permite establecer el origen y el destino de la comunicación.
- Puerto de origen y destino. Permite establecer los puertos origen y destino de la comunicación. Indica el número de puerto o el rango de puertos aplicado a la dirección de red que le precede.
- Dirección. Permite establecer el sentido de la comunicación. Las posibles opciones son:
- ->, <- y <>.
- Acción. Permite indicar la acción que se debe realizar sobre dicho paquete. Los posibles valores son:
  - ✓ alert: Genera una alerta usando el método de alerta seleccionado y posteriormente loggea el paquete.
  - ✓ log: Comprueba el paquete.

- ✓ pass: Ignora el paquete.
- ✓ activate: Alerta y luego activa otra regla dinámica.
- ✓ dynamic: Permanece ocioso hasta que se active una regla, entonces actúa como un inspector de reglas Roesch (2012).

2.2.7.8 Opciones de regla

Las opciones están separadas entre sí, por (;) y las claves de las opciones están separadas por (:). Hay cuatro tipos de opciones:

- Meta-data. Proporciona la información sobre la regla, pero no tenga alguno afecta durante la detección.
- Payload. Busca patrones (firmas) dentro de la carga útil del paquete.
- Non-Payload. Busca patrones dentro de los demás campos del paquete, que no sean carga útil (por ejemplo, la cabecera).
- Post-detection. Permite activar reglas específicas que ocurren después de que se ejecute una regla. (Roesch, 2012).

Ahora veremos las principales opciones de las reglas:

- msg. Informa al motor de alerta que mensaje debe de mostrar. Los caracteres especiales de las reglas como: y; deben de colocarse dentro de la opción msg con el carácter \.
- flow. Se usa junto con los flujos TCP, para indicar qué reglas deberían de aplicarse sólo a ciertos tipos de tráfico.
- content. Permite que Snort realice una búsqueda sensitiva para un contenido específico del payload del paquete.
- referente. Define un enlace a sistemas de identificación de ataques externos, como bugtraq, con id 788.
- classtype. Indica qué tipo de ataques intentó el paquete. La opción classtype, usa las classifications definidas en el archivo de configuración de Snort y que se encuentran en archivos como classification.config.
- La sintaxis del classification.config es:
- <nombre_clase>, <descripción_clase >, <prioridad_por_defecto >.

- La prioridad es un valor entero, normalmente 1 para prioridad alta, 2 para media y 3 para baja.
- La opción classification para el attempted-admin que aparece en classification.config, es la siguiente:
- config classification: attempted-admin Administrator Privilege Gain,1
- La opción sid, en combinación con la opción rev, unicamente identifica una regla Snort, correlacionando el ID de la regla individual con la revisión de la regla. (Roesch, 2012).

2.2.8 PulledPork actualización de reglas automáticas.

Nos encontramos en un tiempo en el cual, cada vez que pasa, se descubren nuevas vulnerabilidades con respecto al software, y quizás alguien con malas intenciones intente aprovecharse de los fallos encontrados, para hacer un nuevo tipo de ataque y que nuestros sistemas informáticos de seguridad no se percaten de dicha intrusión para de esa manera evadir nuestra seguridad. PulledPork es un script escrito en Perl que descarga, combina, instala y actualiza conjuntos de reglas de varios sitios que serán usados por el IDS Snort. Mediante un Oinkcode que se adquiere al registrase en la página oficial de snort gratuitamente podremos adquirir una gran cantidad de reglas para nuestro sistema. (Llopis, 2017).

2.2.9 Barnyard2 optimizando recursos

Barnyard2, es una herramienta para procesar ficheros Unified2 y que permite múltiples configuraciones de salida; la más utilizada, la escritura en Base de datos. Snort requiere muchos recursos para escribir eventos en modo legible para el usuario, ya sea en la consola o en archivos de texto, los llamados log. Idealmente, se quiere que los eventos de Snort se almacenen en una base de datos MySQL para poder ver, buscar y perfilar los eventos. Belda (2011). Para adquirir eventos Snort de modo eficaz en una base de datos MySQL se usa Barnyard2. Se configura Snort para guardar los eventos en formato binario en una carpeta, y posteriormente guardarlos para que Barnyard2 pueda leer esos eventos de manera asíncrona y los inserte en la base de datos MySQL.

2.2.10 Mysql

MySQL es la base de datos de código abierto más famosa de los clientes. Gracias a su beneficio probado, a su fiabilidad y a su facilidad de uso, MySQL se ha transformado en la base de datos líder elegida para las aplicaciones basadas en web y utilizada por propiedades web de perfil alto, como Facebook, Twitter, YouTube y los cinco sitios web principales. (Oracle, 2015).

2.2.11 GUI Interfaz gráfica de usuario para snort

Las interfaces web nos sirven para poder administrar de una manera más amigable y más entendible el tráfico de la red. Snort cuenta con GUIs que puede usar para visualizar el tráfico, mencionaremos algunos de ellos:

- **BASE:** El motor de estudio y seguridad primordial se basó en el viejo código de base de código ACID. La interfaz GUI de ACID que hoy está muerta, y ha durado unos cinco o seis años. No se ha continuado desarrollando activamente desde poco menos desde el 2003. BASE, un cruce del código ACID, retomada donde el autor único lo dejó, agregó muchas nuevas características y lo hizo fácil de usar, en varios idiomas y altamente GUI funcional. Hubo planes para un rediseño de BASE, incluido el formato de base de datos a partir el que se lee, el gerente de proyecto original de BASE, dejó el proyecto y lo entregó a la nueva dirección. Sin embargo, sigue siendo la interfaz gráfica de usuario de Snort más conocido. (Snort, 2011).
- **OSSIM:** “Hecho por AlienVault, significa "Open Source Security Information Management". No solo puede tomar los registros de Snort y mostrarlos en una interfaz de excelente apariencia, sino que también se integra con muchas otras herramientas (p0f, arpwatch, pads, nessus, ntop, nagios, etc.) para una interfaz de usuario consistente. Personalmente, nunca he usado esta herramienta, pero he oído hablar de la gente que la usa, la utiliza y me resulta realmente divertida utilizarla.” (Snort, 2011).
- **SGUIL:** (se pronuncia "Squeel") SGUIL comenzó como la " GUI de S nort de L amers". El proyecto, seguido por Bamm Vischer, es un sistema de varias partes que consta de un "Sensor", un "Servidor" y un "Cliente". SGUIL no solo es una GUI para Snort, sino que también integra otras tecnologías en el registro de datos para uso del analista, incluido la captura de paquetes completos a tiempo completo. (Snort, 2011).

- **SNORBY:** “Snorby es una aplicación web (front-end) basada en Ruby on Rails, que interactúa con un IDPS para monitorizar gráficamente la seguridad de la red. Se comunica con sistemas como Snort, Sagan, Suricata y cualquier otro que genere eventos de log en el formato binario Unified2.” (Snort, 2011).

2.2.12 Pentesting

“Pentesting o Penetration Testing es la práctica de atacar diversos entornos con la intención de descubrir fallos, vulnerabilidades u otros fallos de seguridad, para así poder prevenir ataques externos hacia esos equipos o sistemas”. La prueba de penetración (también llamada prueba de lápiz) es la práctica de probar un sistema informático, red o aplicación web para encontrar vulnerabilidades que un atacante podría explotar (OpenWebinars, 2015, p. 4).

2.2.13 DDoS

DDoS significa denegación de servicio distribuido. Este es un tipo de ataque digital que intenta separar un sitio o, en cualquier caso, hacerlo más lento que los invitados. Esto se logra sumergiendo el sitio con tráfico nocivo, independientemente de si está coordinado con el sistema o con el servidor. Este es un tráfico maligno conocido como botnet, que es un sistema de dispositivos que los agresores han contaminado con malware para controlarlos desde un lugar remoto. Los ataques DDoS son inmensamente convincentes para hacer daño a un sitio o negocio. ¿Cómo funciona? Cuando no pueden acceder a un sitio o administración. Esta expresión no puede ser utilizada para perder dedicación o duda (Servidorinfo, 2018, p. 6).

2.2.13.1 Tipos de ataque DDoS

Existen varios tipos de ataques DDoS, mencionaremos cuales son los más conocidos/comunes.

- **Inundación UDP.** Es un asalto en el que un servidor de víctimas desafortunado es dominado por solicitudes que parecen ser reales. En él, los datagramas UDP se envían a puertos irregulares en el servidor de objetivos. El servidor en ese punto intenta coordinar estos datagramas con una aplicación en el puerto de obtención, en cualquier caso, ya que estas aplicaciones no se mantienen firmes para las solicitudes,

no pueden combinarse y el servidor debe reaccionar a la solicitud diciendo que objetivo no es accesible (Servidorinfo, 2018, p. 6).

- **Inundación del DNS.** Otro ataque que ve a un servidor individual lesionado se ve obstaculizado por solicitudes que parecen ser genuinas. En una inundación de DNS, el servidor DNS está asediado por demandas UDP. Un servidor de marco de nombres de área o DNS se encarga de realizar una interpretación del nombre del espacio a su dirección IP para que su dispositivo pueda alcanzar ese espacio a través de Internet, de modo que cuando un servidor DNS se encuentre con la gestión de las falsas demandas de una inundación de DNS, no tiene recursos accesibles para dirigir a clientes reales al sitio al que intentan acceder (Servidorinfo, 2018, p. 6).
- **Inundación HTTP.** Es una especie de asalto DDoS que incluye enviar solicitudes HTTP a un servidor que requiere que el servidor asigne la mayor medida de activos a su reacción. Estas solicitudes son frecuentemente demandas HTTP-POST que son demandas de sustancias dinámicas que deben ser producidas por el servidor. Los agresores completan regularmente un pequeño compromiso con su objetivo en una inundación de HTTP para determinar qué activos o sustancias requieren la preparación más seria del servidor, de esta manera gastando la mayor parte de los activos en cada solicitud (Servidorinfo, 2018, p. 6).
- **SYN flood.** En este tipo de inundación, un sistema bot de asalto llega al desafortunado servidor de víctimas con una enorme cantidad de solicitudes o sincronización SYN, la fase inicial en la configuración de una asociación entre el programa y el servidor del sitio. El servidor en ese punto reacciona a las demandas de SYN con el SYN-ACK normal, dejando un puerto abierto para cada una de las solicitudes. El programa o el lado asaltante nunca reacciona a estas reacciones, dejando los puertos abiertos mientras el servidor lucha para mantenerse al tanto del número considerable de respuestas que necesita enviar, sin dejar nada para los clientes reales, (Servidorinfo, 2018, p. 6).
- **Amplificación DNS.** No confunda esto con una inundación de DNS. En la amplificación de DNS, los atacantes envían solicitudes a varios solucionadores de DNS disponibles públicamente, con solicitudes falsificadas para que parezca que provienen de la víctima. Todos los solucionadores de DNS responden a las solicitudes utilizando la IP de la víctima, golpeándola con un gran número de respuestas a la vez. Esto se denomina ataque de reflexión y se puede amplificar

(como sugiere su nombre) mediante el envío de pequeñas solicitudes que requieren grandes respuestas de los resolutores de DNS, (Servidorinfo, 2018, p. 6).

- **Amplificación NTP.** Los servidores NTP son servidores comunitarios que permiten que los dispositivos asociados con Internet sincronicen sus cronometradores. Estos servidores también permiten demandas de datos sobre los últimos 600 hosts asociados con el servidor. Los agresores hacen un mal uso de esto al enviar estas solicitudes desde una IP falsa, por lo que da la impresión de que se originó en la persona en cuestión, lo que hace que los servidores NTP reaccionen con estos registros gigantescos, que dominan los activos del servidor (Servidorinfo, 2018, p. 6).

2.2.14 ISO 27001

Existen diversas metodologías para la gestión de la seguridad de la información, así como MAGERIT que básicamente se detalla en la norma ISO 27005, OCTAVE que consta de 3 fases, evaluación, identificación y desarrollo de un plan estratégico. Estas metodologías nos indican los pasos a seguir para su correcta ejecución. Según lo conceptuado en la evaluación de riesgos, la metodología para evaluar todo el procedimiento que se está usando es de la ISO 27001 el cual se plantea así en la Figura 3. "El estándar ISO/IEC 27001 proviene de una familia de normas llamada ISO/IEC 27000, la cual tiene la intención de ayudar a las organizaciones de todo tipo y tamaño a implementar y operar un SGSI, y consta de varias normas internacionales bajo el título de tecnologías de la información – técnicas de seguridad" (Advicera, 2016, p. 7).

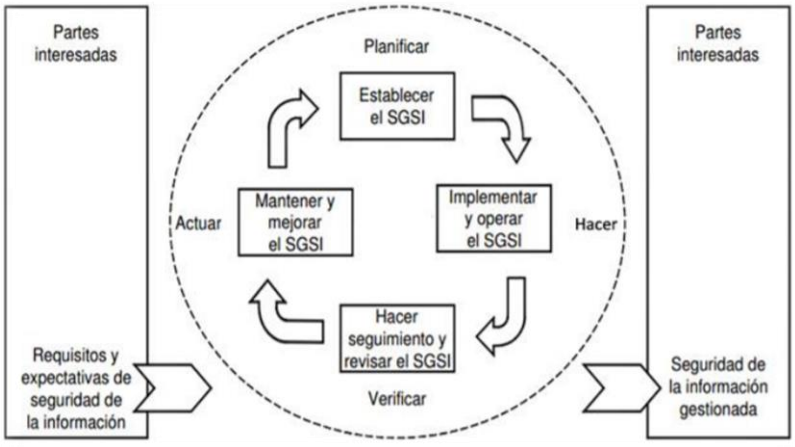


Figura 3. Modelo PHVA aplicado a los procesos de los SGSI
Fuente: Organización internacional de estandarización (2018).

Elementos o fases para la implementación de un SGSI. El Sistema de Gestión de La Seguridad de la Información que propone la Norma ISO 27001 se puede resumir en las siguientes fases que se detallan en la figura 4.

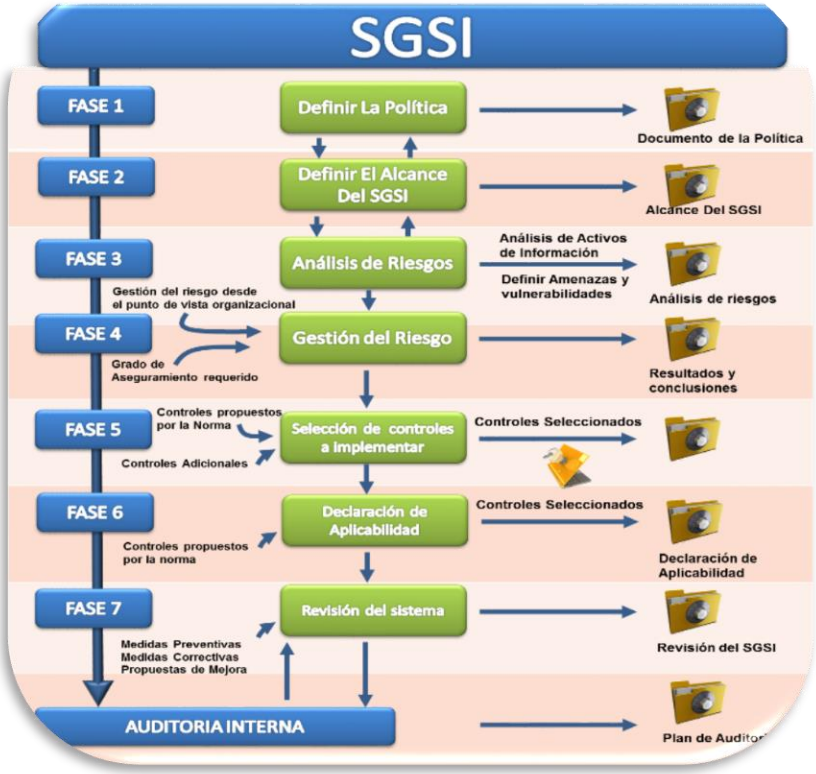


Figura 4. Elementos, fases de la implementación de la SGSI.
Fuente: Normas ISO (2015).

2.2.15 IDS/IPS Cumpliendo con la norma ISO 27001

En diferentes eventos, hemos discutido los controles del sistema que dependen del estándar ISO 27001, ya que hemos discutido los servidores de seguridad y la desagregación del sistema. A pesar del hecho de que estas alternativas mejoran la seguridad del sistema, crean un impacto básico: los controles identificados con la periferia, que no pueden disparar en las ocasiones que ocurren dentro de las zonas de aseguramiento, (SGSI, 2016).

"Para mejorar la eficiencia del NIDS, una empresa debe tener en cuenta que el NIDS se colocará en la red, y debe conocer todos los métodos de evaluación de tráfico y los modos en los que se utilizarán. La norma ISO 27001 es una gran aliada en este tema." (SGSI, 2016).

Un NIDS debe configurarse en un punto donde pueda controlar el tráfico hacia y desde todos los dispositivos en el sistema. Algunos lugares propuestos son cortafuegos y servidores básicos. Debe transmitirse un NIDS para distinguir las interrupciones más ampliamente reconocidas. Con respecto a los servidores, un NIDS puesto allí puede ofrecer la ventaja de los ejemplos de tráfico explícitos para configurar sus configuraciones y distinguir incluso las interrupciones más discretas, (SGSI, 2016).

2.3 Marco conceptual

2.3.1 IDS

Según Tiwari (2011) "IDS es una evolución que mejora la seguridad de la red. Y salvaguarda los datos de la organización. El IDS ayuda al administrador de la red para detectar cualquier actividad maliciosa en la red y alerta al administrador para obtener los datos seguros tomando las acciones apropiadas contra esos ataques. Una intrusión se refiere a cualquier acceso no autorizado o malicioso utilizando los recursos de información. Un intruso o un atacante es una entidad del mundo real que trata de encontrar un medio para ganar el acceso no autorizado a la información, causando daños o participando en otras actividades maliciosas."

2.3.2 IPS

Según Roesch (2010) "El término Sistema de Prevención de Intrusión IPS, es usado para combinar o unificar los conceptos de sistema de detección y de sistema de prevención. Es importante tener en cuenta que la definición sólo se ajusta a ataques conocidos. Así pues, un sistema de prevención de intrusos inline o de red es cualquier dispositivo de hardware o software que tiene la capacidad de detectar y prevenir ataques conocidos."

Un IPS nos ayudan a no permitir acciones maliciosas usando diferentes algoritmos que trabaja el bloqueo basado en firmas de tráfico y/o ataques conocidos.

2.3.3 Snort

Según Roesch (2012) "Snort es un sistema de detección de intrusiones basado en red (NIDS). Su funcionamiento es similar al de un sniffer ya que monitoriza todo el tráfico de la red en búsqueda de cualquier tipo de intrusión. Implementa un motor de detección de

ataques y barrido de puertos que permite registrar, alertar y responder ante cualquier anomalía previamente definida como patrones."

Snort es uno de los motores más usados en cuanto a detección de intrusos y trabaja bajo licencia GPL, es gratuito y está disponible para las plataformas Windows y GNU/Linux.

2.3.4 Barnyard2

Según Belda (2011) "Barnyard2 es una herramienta para procesar ficheros Unified2 que permite múltiples configuraciones de salida; la más utilizada, la escritura en Base de datos."

Barnyard2 es un intérprete para las alertas que va recopilando snort, nos ayuda a convertir las alertas en un lenguaje legible para que se pueda almacenar en una base de datos MySQL. Una vez guardada en base de datos ya podemos hacer diferentes estudios de las alertas que se tienen tanto en días, horas, meses o años.

2.3.5 PulledPork

Según Llopis (2017) "PulledPork es un script escrito en Perl que descarga, combina, instala y actualiza conjuntos de reglas de varios sitios que serán usados por el IDS Snort."

PulledPork nos brinda las funcionalidades de descarga de reglas automática de conjunto de reglas, generación del archivo sid-msg.map para el reconocimiento de la DB, compatibilidad con las reglas del IDS Suricata.

2.3.6 Snorby

Snorby es una aplicación web de Ruby on Rails para la seguridad del sistema que se interconecta con los marcos de identificación de interloper convencionales actuales (Snort, Suricata y Sagan). Las ideas esenciales que impulsan a Snorby son la facilidad, la asociación y el poder. Snorby es una aplicación gratuita, de código abierto y muy enfocado para la observación de sistemas para uso privado y comercial.

2.3.7 Norma ISO 27001

Según ISOTools (2016) ISO 27001 es un estándar global que permite la confirmación, el secreto y la respetabilidad de la información y los datos, al igual que los

marcos que procesan. La norma ISO 27001: 2013 para Sistemas de Gestión de Seguridad de la Información permite a las asociaciones estudiar el peligro y aplicar los controles fundamentales para aliviarlos o eliminarlos.

CAPÍTULO III. Materiales y métodos

3.1 Lugar de ejecución

La presente investigación se aplicó en el área de redes y conectividad del departamento de Dirección de tecnologías de la información de la Universidad Peruana Unión – Filial Juliaca. Para el monitoreo de algunos sistemas propios de la institución tales como Alfresco, Pagina web C.A.T., Eventos y Encuestas.

3.2 Materiales

- Recursos Humanos: Jefe de proyecto, Jefe de redes del departamento de DTI, Jefe de Servidores del departamento de DTI.

Tabla 3

Materiales, software y hardware

Software y hardware	Cantidad
PC o Laptop - Capacidades básicas	1
S.O. Kali Linux 64bits	1
S.O. Windows (Con licencia) 64 bits	1
Tarjetas de red Intel	4
Servidor - Dell Xeon - 16 GB Ram	1
S.O. Ubuntu Server 16.04	1
Ssh Putty	1
Internet	-
Electricidad	-
USB	1
PulledPork	-
Barnyard2	-
VMware Esxi 6.5	-
Mysql	-
Snorby	-
Otros gastos	

Fuente: Elaboracion propia, (2018).

3.3 Metodología

3.3.1 Investigación aplicada

Esta investigación es de tipo aplicada ya que nos basaremos en los hallazgos tecnológicos para de esa manera generar el conocimiento con la aplicación directa a los problemas de la institución. Investigación Aplicativo Según Chavarriaga, Arboleda, & Lidis, (2004) es la etapa en donde se busca madurar la tecnología definida en la etapa inicial, trabajando en el desarrollo de la tecnología y en su aplicación en situaciones reales. A medida que se tienen experiencias de aplicación de la tecnología por parte del grupo de investigación y empresas de la industria, se encuentran las posibles fallas en el modelo de solución propuesto y se definen las adaptaciones que puedan ser requeridas para su aplicación en empresas.

3.3.2 Arquitectura de solución

La arquitectura se explica de la siguiente manera. Internet llega a través del equipo administrable fortigate el cual asigna una red en este caso la red 192.168.96.0, luego pasa por un switch a nuestro servidor Snort. Ahora nuestro IDS será el centro de todo, si un servidor quiere conectarse a internet parsara por snort verificara el trafico y seguirá el proceso, si un usuario quiere conectarse a un servidor primero pasar por snort si tiene acceso autorizado la conexión será exitosa, si un usuario quiere acceder a internet tambien primero pasara por snort y llegara a internet si el trafico es bueno la conexión será exitosa.

Basado en la norma ISO 27001, donde nos recomienda colocar el IDS en un punto donde pueda monitorizar el tráfico. Y cumpliento las 4 etapas de planificar, implementar, medir y mejorar.

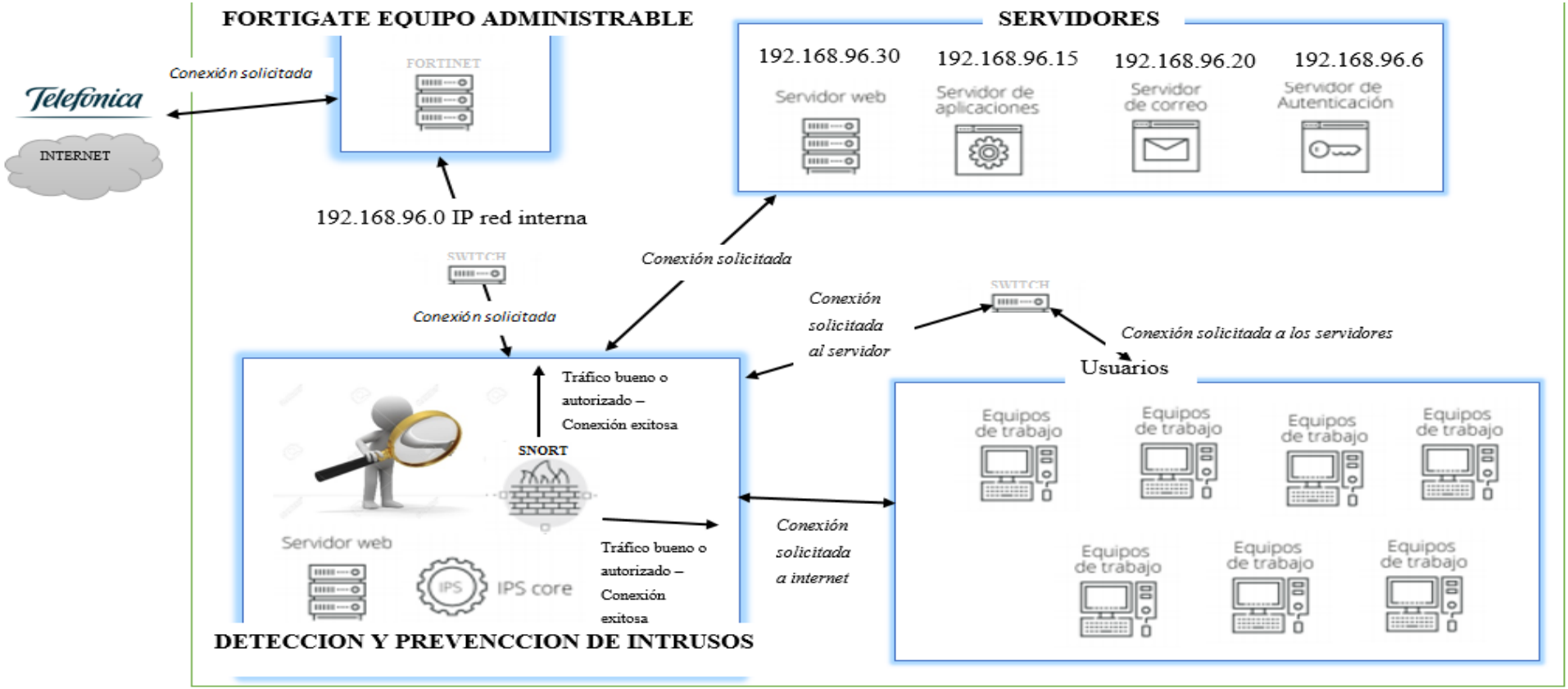


Figura 5. Arquitectura de solución.
Fuente: Elaboracion propia (2018).

3.3.3 Diseño de la implementación con la metodología de la investigación

Según lo conceptuado en la evaluación de riesgos, la metodología para evaluar todo el procedimiento que se uso es de la ISO 27001 el cual se plantea así en la Figura 3.

El enfoque basado en las etapas del sistema de gestión de seguridad de la información, en este proyecto se reflejó en 4 etapas, y en la estructura se vera de la siguiente manera. Planificar, Implementar, Medir y Mejorar.

3.3.4 ISO 27001

3.3.4.1 Planear

Se estableció el objetivo del análisis y diseño de las reglas según las amenazas más frecuentes mostrados por el fortigate, para mejorar la seguridad de la información de la organización, como también la evaluación de reglas globales existentes para el monitoreo del tráfico.

3.3.4.2 Implementar

Se estableció el objetivo de la instalación y configuración del motor de amenazas IDS/IPS snort bajo los siguientes módulos.

3.3.4.2.1 Módulo configuración del servidor y S.O.

Para la instalación y configuración de nuestro servidor se usó la plataforma de virtualización VMware Esxi 6.5. El cual ya tiene configurado el área de redes en la Universidad Peruana Unión – Filial Juliaca. ESXi es un hipervisor de tipo 1, lo que significa que se ejecuta directamente en el hardware del sistema sin la necesidad de un sistema operativo (S.O.). Los hipervisores de tipo 1 también se conocen como hipervisores de metal desnudo porque se ejecutan directamente en el hardware. Ya que se contó con una plataforma virtualizada se eligió el S.O. Ubuntu Server 16-04 por la eficacia y la estabilidad que esta posee.

3.3.4.2.2 Módulo de configuración de snort en modo NIDS

Para la instalación de Snort se necesitó ciertos requisitos previos para que Snort pueda funcionar correctamente. Fueron necesarias las librerías (libpcap-dev), (libpcre3-dev) y (libdumbnet-dev). De igual manera se necesitó hacer uso de snort DAQ que es la biblioteca de adquisición de datos para snort.

Se hizo uso de varias librerías que fueron requisitos fundamentales para el funcionamiento de snort entre ellas tenemos el paquete de luajit v5.1 el cual fue necesario al momento de instalar snort. Se configuró snort en modo promiscuo para que pueda monitorear toda la red.

Para que snort pueda iniciarse junto con el S.O. se procedió a crear un archivo demonio indicando los parámetros y la interfaz de red que se va a escanear.

3.3.4.2.3 Módulo de configuración de snort en modo IPS

Para la configuración de snort en modo IPS o dicho de otro modo en modo inline el cual se trabajó con afpacket, es necesario instalar la librería de NFQ una vez ya instalada dicha libreria se procedio a instalar configurar el archivo snort.conf, es muy importante ingresar las líneas: config daq = afpacket y config daq_mode = inline, de esta manera le estamos diciendo a snort que corra en modo IPS.

Para snort en modo IPS tenemos ciertas acciones adicionales para las reglas estas son:

Drop: bloquea y registra el paquete.

Reject: bloquea el paquete, lo registra y luego envía un reinicio de TCP si el protocolo es TCP o un mensaje de puerto ICMP inalcanzable si el protocolo es UDP.

Sdrop: bloquea el paquete pero no lo registra.

3.3.4.2.4 Módulo de configuración de Barnyard2.

Para poder usar Barnyard2 es necesario instalar algunas librerías como libtool e instalar mysql. Se procedió a configurar barnyard2.conf para ingresar el usuario la contraseña y el nombre de la base de datos. Como también se cambió el permiso de lectura

de dicho archivo. Como anteriormente se mencionó que Barnyard2 nos ayuda en la optimización de recursos, se optó por usar Barnyard2 y crear una base de datos llamada snort para que todo el tráfico encontrado se pueda almacenar en la base de datos mysql y de esa manera ver, buscar y perfilar eventos.

Para que barnyard2 pueda cargar los datos en la base datos se procedió a configurar un archivo demonio que pueda iniciarse junto con el S.O.

3.3.4.2.5 Módulo de configuración de Pulledpork

Al instalar snort viene con algunas reglas por defecto de instalación y ya que nosotros no quisimos usar esas reglas optamos por usar pulledpork y para poder adquirir esas reglas se necesitó adquirir un oinkcode creándonos una cuenta en la página oficial de snort. Y de esa manera obtuvimos una mayor cantidad de reglas como también de la comunidad de snort, para poder hacer uso de estas reglas se procedió a configurar el archivo pulledpork.conf e introducir el oinkcode y editando algunas líneas que son necesarias para el funcionamiento de la descarga de pulledpork. Una vez terminada la configuración para pulledpork, no se creó un archivo demonio si no, se procedió a crear un crontab para que a diario pueda obtener las reglas.

3.3.4.2.6 Módulo de configuración de snorby

Para poder instalar snorby fue necesario la instalación de apache y ruby en la versión 2.3 como también de algunas gemas necesarias como bundler, rails y rake. Para que snorby pueda conectarse a la base de datos se procedió a crear una base de datos con el nombre snorby y una contraseña una vez creado nuestra base de datos se procedió a configurar el archivo database.yml estando en ese archivo se introdujo el usuario y la contraseña de la base de datos creada. Se instaló el módulo phusion passenger que es un servidor de aplicaciones para apache y de esa manera lanzar snorby.

Se configuró el módulo passenger y se creó el sitio web para snorby poniendo snorby en el puerto 80, para que snorby pueda conectarse a nuestra base de datos de barnyard2 se configuro el archivo barnyard2.conf e introducir la base de datos el cual se almacena el trafico guardado.

De igual manera se procedió a crear un demonio para que pueda iniciarse cuando el S.O. encienda. Una vez configurado todo y entrando en la interfaz web se procedió a crear un usuario y contraseña para el logueo en snorby.

3.3.4.3 Medir

Se estableció el objetivo del monitoreo y el análisis del tráfico de datos mediante los reportes que snorby nos brinda a través de la interfaz web.

3.3.4.4 Mejorar

Se estableció el objetivo de pruebas del análisis del tráfico. Se tomó acciones correctivas y preventivas, haciendo las pruebas con amenazas desarrolladas por el usuario, para medir la calidad y mostrar los resultados. Según el resultado mostrado se realizó la medición del sistema, como también dándole una auditoria al sistema ya una vez implementado.

CAPÍTULO IV. Desarrollo del sistema

4.1 Análisis y diseño de reglas – Planificación ISO 27001

Para la primera fase de nuestra metodología tendremos el análisis de las reglas según los reportes del tráfico malicioso del equipo fortigate que la Universidad Peruana Unión – Juliaca cuenta. Una vez analizada vemos que en la categoría de tráfico malicioso tenemos como sitios web maliciosos, ataques de inyección como también evadir el control del firewall entre otros. Anexo C - H. viendo este reporte empezamos a investigar qué tipo de reglas son las que pueden bloquear este tipo de tráfico malicioso. Nos guiamos de Quadrant (2019) para las reglas propuestas.

A continuación mostramos cuales fueron las reglas que se propusieron.

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Not Suspicious Traffic"; program: snort; content: "Classification|3a| Not Suspicious Traffic"; classtype: not-suspicious; normalize; sid: 5000976; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Unknown Traffic"; program: snort; content: "Classification|3a| Unknown Traffic"; classtype: unknown; normalize; sid: 5000977; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Bad Traffic"; program: snort; content: "Classification|3a| Bad Traffic"; classtype: bad-unknown; normalize; sid: 5000978; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Attempted Information Leak"; program: snort; content: "Classification|3a| Attempted Information Leak"; classtype: attempted-recon; xbits: set,recon,track ip_src, expire 86400; normalize; sid: 5000979; rev:6;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Information Leak"; program: snort; content: "Classification|3a| Information Leak"; classtype: successful-recon-limited; xbits: set,recon,track ip_src, expire 86400; normalize; sid: 5000980; rev:6;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Large Scale Information Leak"; program: snort; content: "Classification|3a| Large Scale Information Leak"; classtype: successful-recon-largescale; xbits: set,recon,track ip_src, expire 86400; normalize; sid: 5000981; rev:6;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Attempted Denial of Service"; program: snort; content: "Classification|3a| Attempted Denial of Service"; classtype: attempted-dos; normalize; sid: 5000982; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Denial of Service"; program: snort; content: "Classification|3a| Denial of Service"; classtype: successful-dos; normalize; sid: 5000983; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Attempted User Privilege Gain"; program: snort; content: "Classification|3a| Attempted User Privilege Gain"; classtype: attempted-user; normalize; sid: 5000984; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Unsuccessful User Privilege Gain"; program: snort; content: "Classification|3a| Unsuccessful User Privilege Gain"; classtype: unsuccessful-user; normalize; sid: 5000985; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Successful User Privilege Gain"; program: snort; content: "Classification|3a| Successful User Privilege Gain"; classtype: successful-user; normalize; sid: 5000986; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Attempted Administrator Privilege Gain"; program: snort; content: "Classification|3a| Attempted Administrator Privilege Gain"; classtype: attempted-admin; normalize; sid: 5000987; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Successful Administrator Privilege Gain"; program: snort; content: "Classification|3a| Successful Administrator Privilege Gain"; classtype: successful-admin; normalize; sid: 5000988; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Decode of an RPC Query"; program: snort; content: "Classification|3a| Decode of an RPC Query"; classtype: rpc-portmap-decode; normalize; sid: 5000989; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Executable code was detected"; program: snort; content: "Classification|3a| Executable code was detected"; classtype: shellcode-detect; normalize; sid: 5000990; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] A suspicious string was detected"; program: snort; content: "Classification|3a| A suspicious string was detected"; classtype: string-detect; normalize; sid: 5000991; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] A suspicious filename was detected"; program: snort; content: "Classification|3a| A suspicious filename was detected"; classtype: suspicious-filename-detect; normalize; sid: 5000992; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] An attempted login using a suspicious username was detected"; program: snort; content: "Classification|3a| An attempted login using a suspicious username was detected"; classtype: suspicious-login; normalize; sid: 5000993; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] A system call was detected"; program: snort; content: "Classification|3a| A system call was detected"; classtype: system-call-detect; normalize; sid: 5000995; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] A TCP connection was detected"; program: snort; content: "Classification|3a| A TCP connection was detected"; classtype: tcp-connection; normalize; sid: 5000996; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] A Network Trojan was detected"; program: snort; content: "Classification|3a| A Network Trojan was detected"; classtype: trojan-activity; normalize; sid: 5000997; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] A client was using an unusual port"; program: snort; content: "Classification|3a| A client was using an unusual port"; classtype: unusual-client-port-connection; normalize; sid: 5000998; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Detection of a Network Scan"; program: snort; content: "Classification: Detection of a Network Scan"; classtype: network-scan; normalize; sid: 5000999; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Detection of a Denial of Service Attack"; program: snort; content: "Classification|3a| Detection of a Denial of Service Attack"; classtype: denial-of-service; normalize; sid: 5001000; rev:5;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Suspected TCP injection"; flow:to_client; window:1; fragbits:!D;sid:1000000; rev:2;)

alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"Data Served by known Injection Device (PerfTech)"; flow:to_client,established; content:"PerfTech"; http_header; sid:1000001; rev:2;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Detection of a non-standard protocol or event"; program: snort; content: "Classification|3a| Detection of a non-standard protocol or event"; classtype: non-standard-protocol; normalize; sid: 5001001; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Generic Protocol Command Decode"; program: snort; content: "Classification|3a| Generic Protocol Command Decode"; classtype: protocol-command-decode; normalize; sid: 5001002; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] access to a potentially vulnerable web application"; program: snort; content: "Classification|3a| access to a potentially vulnerable web application"; classtype: web-application-activity; normalize; sid: 5001003; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Web Application Attack"; program: snort; content: "Classification|3a| Web Application Attack"; classtype: web-application-activity; normalize; sid: 5001004; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Misc activity"; program: snort; content: "Classification|3a| Misc activity"; classtype: misc-activity; normalize; sid: 5001005; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Misc Attack"; program: snort; content: "Classification|3a| Misc Attack"; classtype: misc-attack; normalize; sid: 5001006; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Generic ICMP event"; program: snort; content: "Classification: Generic ICMP event"; classtype: icmp-event; normalize; sid: 5001007; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] SCORE! Get the lotion! [Porn]"; program: snort; content: "Classification|3a| SCORE! Get the lotion!"; classtype: kickass-porn; normalize; sid: 5001008; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Potential Corporate Privacy Violation"; program: snort; content: "Classification|3a| Potential Corporate Privacy Violation"; classtype: policy-violation; normalize; sid: 5001009; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Attempt to login by a default username and password"; program: snort; content: "Classification|3a| Attempt to login by a default username and password"; classtype: default-login-attempt; normalize; sid: 5001010; rev:5;)

4.2 Instalación y configuración de snort – Implementación ISO 27001

Previo a la configuración de snort tenemos que configurar nuestro servidor, especificamos los datos que nos pida, como el nombre que en nuestro caso se llamara Snort y en cuanto a las características del servidor se optó por darle la siguiente configuración: memoria ram un total de 4GB, disco duro de 16 GB, 2 tarjetas de red según en la red de nuestro servidor también especificamos porque medio se realizará la instalación del S.O. que en nuestro caso se realizará mediante un archivo ISO. Tal como nos muestra en la Figura 6.

En cuanto al S.O. se instaló el S.O. Ubuntu Server 16.04 se inició con la selección de nuestro archivo ISO y se procedió con la instalación. Tal como nos muestra en la Figura 7.

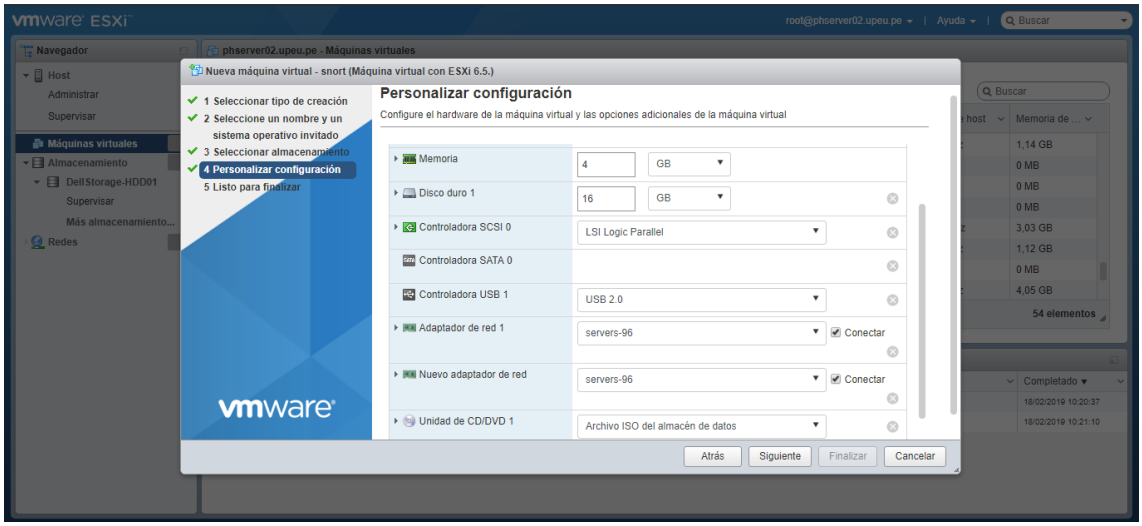


Figura 6. Configuración del servidor en VMware
Fuente: Elaboración propia (2018).

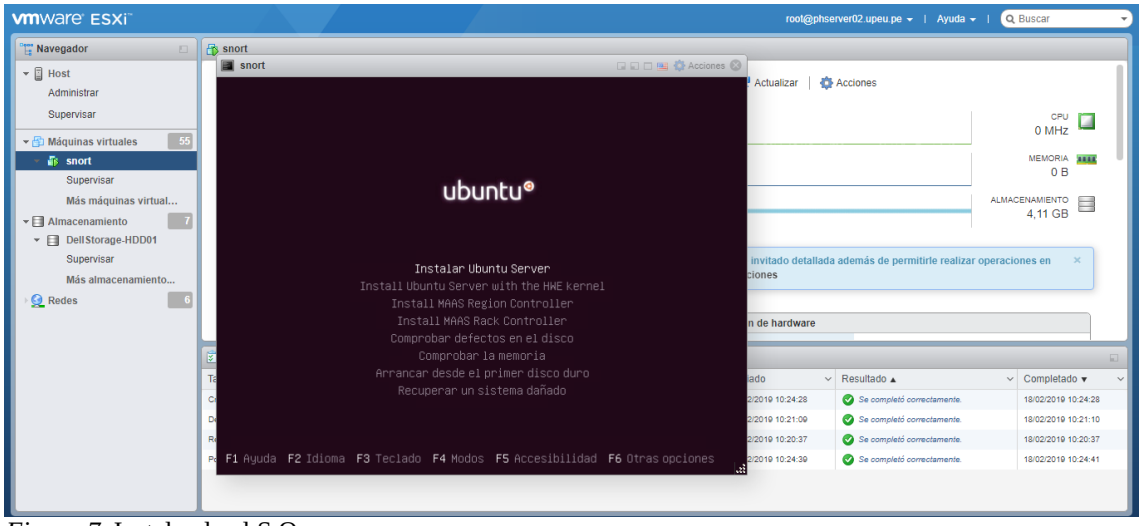


Figura 7. Instalando el S.O.
Fuente: Elaboración propia (2018).

Una vez terminada la instalación y configuración de nuestro servidor y S.O. se procedió a iniciar sección en nuestro servidor y a continuación se hizo la siguiente configuración:

- sudo apt-get update
- sudo apt-get dist-upgrade -y
- sudo apt-get install -y openssh-server
- sudo reboot

Se usó Update para ver que el sistema este actualizado y se instaló openssh para poder usar el sistema de forma remota y por último se reinició nuestro sistema para que los cambios surtan efecto.

4.3 Configuración tarjeta de red

Se configuro la tarjeta de red ya que por defecto al instalarse viene con la descarga de recepción grande (GRO) y genérica (LRO) si estas características están habilitadas puede que cause algún tipo de problema en la configuración de nuestro IDS/IPS Snort, para poder desactivar el GRO y LRO se hizo uso del comando ethtool: sudo apt-get install –y ethtool una vez instalada se hizo la desactivación respectiva en el archivo /etc/network/interfaces de nuestra red correspondiente. Tal como nos muestra la Figura 8.

Una vez configurado el archivo interfaces se procedió al reinicio de nuestra red y se procedió a la verificación de la des habilitación de GRO y LRO. Tal como nos muestra la Figura 9.

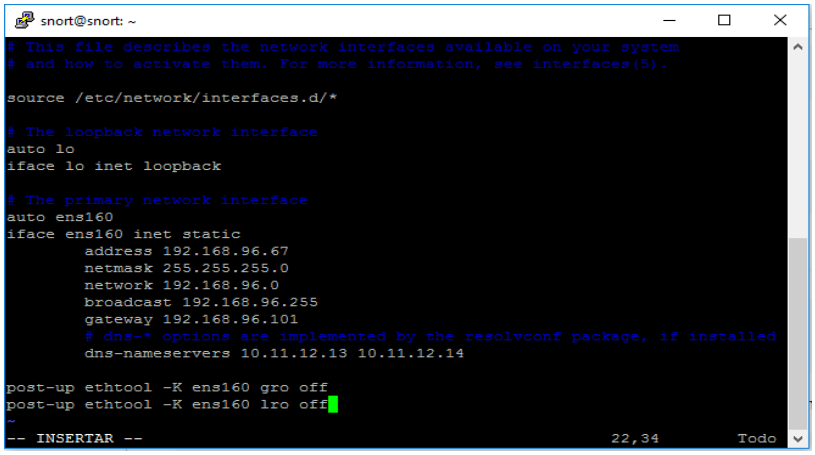


Figura 8. Configuración de archivos de interfaces.
Fuente: Elaboración propia (2018).

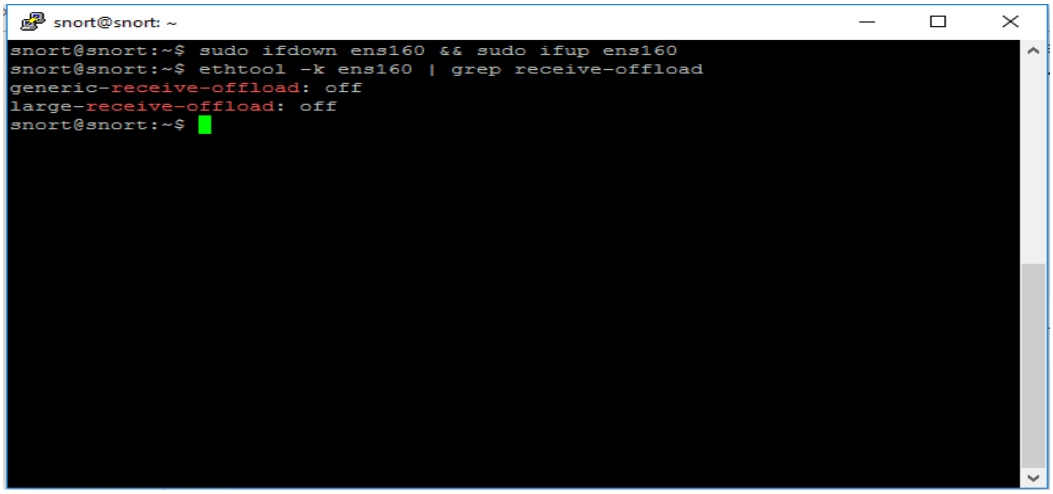


Figura 9. Generic y Large desactivadas.
Fuente: Elaboración propia (2018).

4.4 Librerías previas a instalar snort

Para la instalación de Snort se necesitó ciertos requisitos previos para que Snort pueda funcionar correctamente. Fueron necesarias las librerías (libpcap-dev), (libpcre3-dev) y (libdumbnet-dev). Tal como nos muestra la Figura 10.

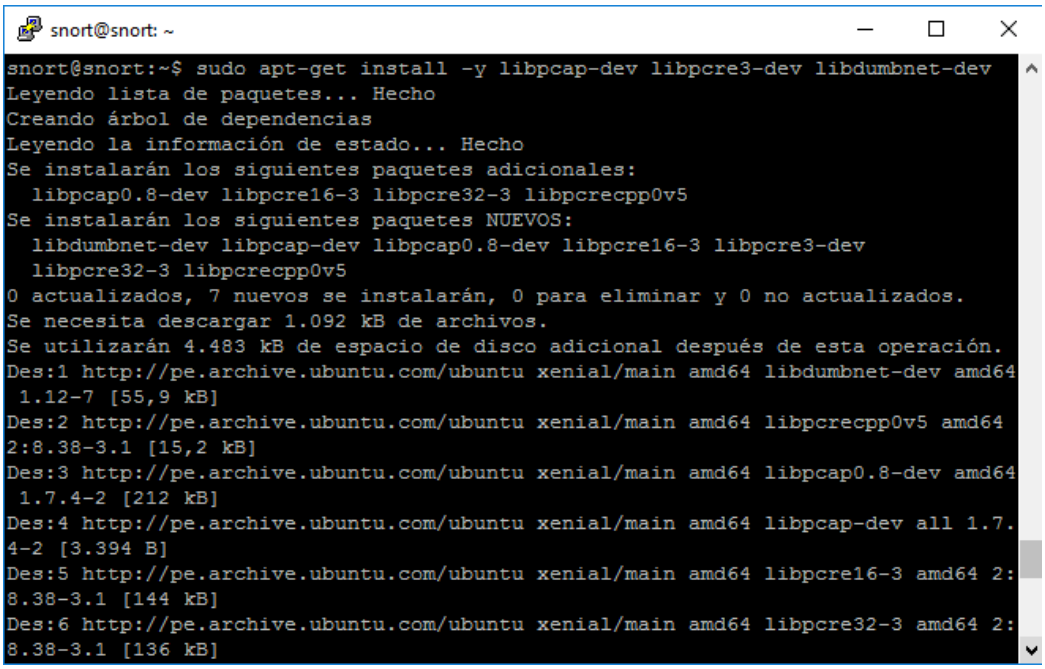


Figura 10. Requisitos previos para instalar snort.
Fuente: Elaboración propia (2018).

Para poder trabajar de una manera más ordenada y tener todos nuestros paquetes de instalación de un solo lugar se creó una carpeta llamada snort_src, para poder hacer uso de Snort DAQ que es la biblioteca de adquisición de datos se instaló un requisito previo: sudo

apt-get install –y bison flex una vez instalada se procedió con la descarga e instalación de SnortDAQ de su última versión 2.0.6 de la página oficial de snort.org. Tal como nos muestra la Figura 11.

- cd ~/snort_src
- wget https://www.snort.org/downloads/snort/daq-2.0.6.tar.gz
- tar -xvzf daq-2.0.6.tar.gz
- cd daq-2.0.6
- ./configure
- make
- sudo make install

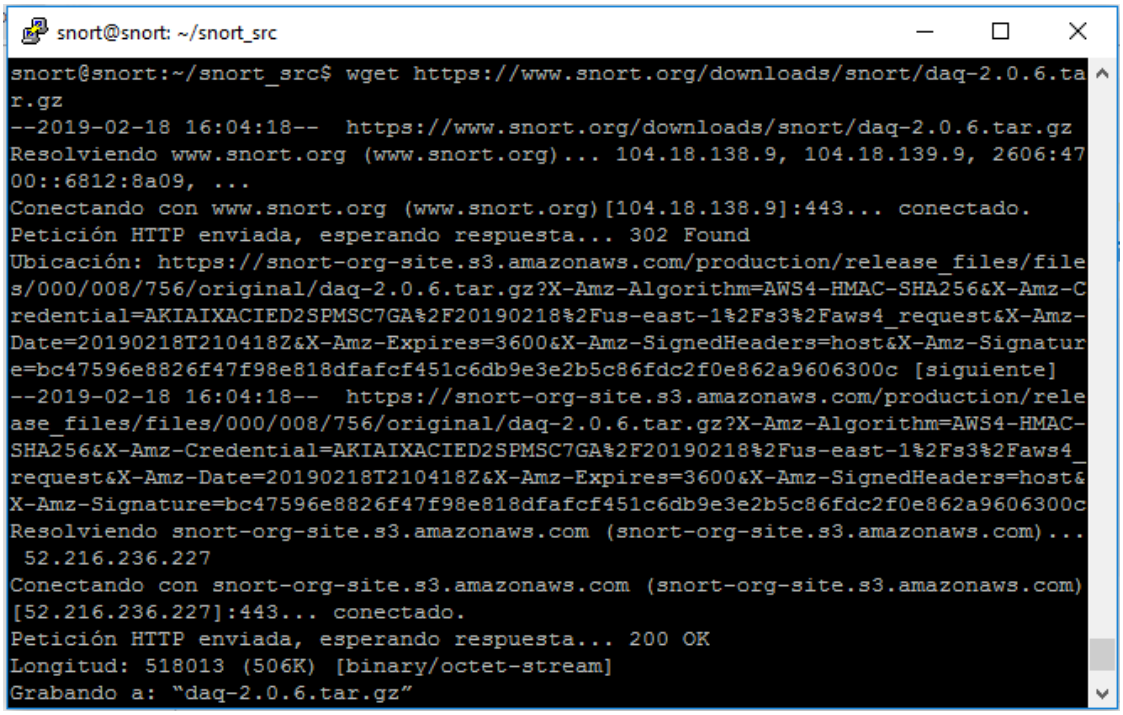


Figura 11. Descarga de la biblioteca Snort Daq.
Fuente: Elaboración propia (2018).

Se necesitó tres requisitos adicionales opcionales para la mejor funcionabilidad de Snort liblzma-dev para la descompresión de archivos swf, openssl y libssl-dev para que nos brinden firmas de archivo SHA y MD5. Tal como nos muestra la Figura 12.

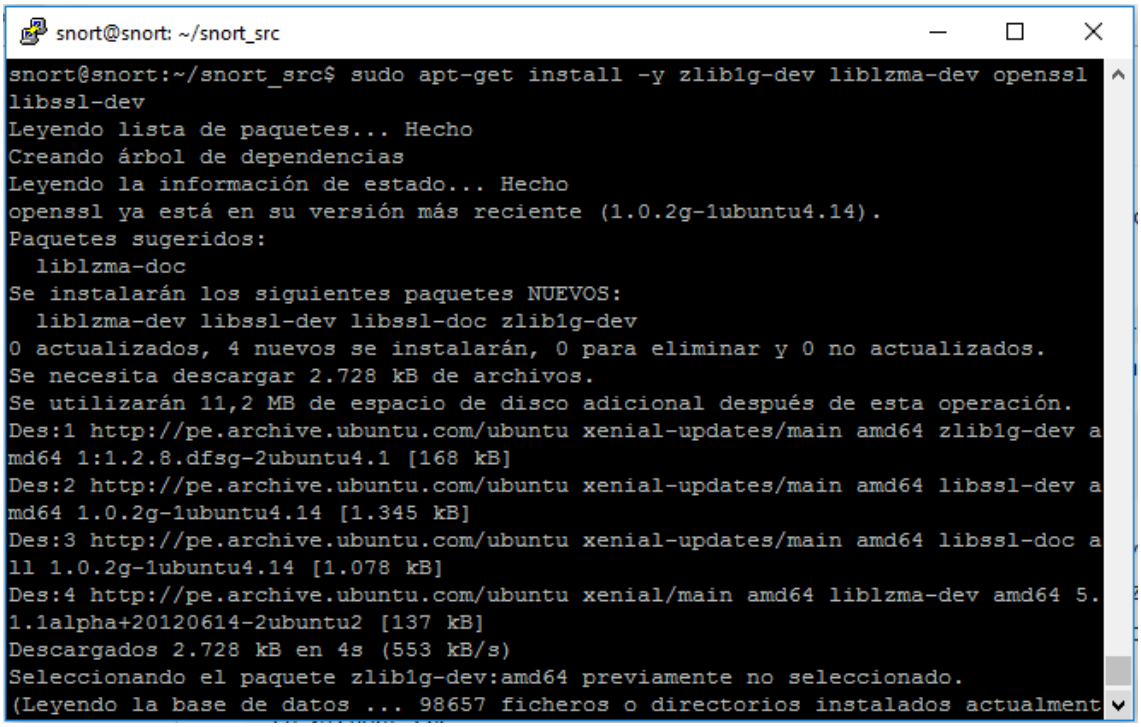


Figura 12. Requisitos Adicionales para configurar Snort.
Fuente: Elaboración propia (2018).

Al ya tener instalados todos los requisitos para el uso de Snort se procedió a la descarga, instalación y configuración de Snort, descargamos Snort de la página oficial snort.org procedemos a descomprimir y hacer uso de la opción –enable-sourcefire que nos permite realizar un monitoreo del rendimiento de las reglas. Tal como nos muestra la Figura 13. Se tuvo un error al hacer uso de la opción –enable-sourcefire que nos decía que necesitaba el uso de LuaJit se solucionó al instalar dicho paquete: sudo apt-get install -y libluajit-5.1-dev pkg-config openssl libssl-dev.

- cd ~/snort_src
- wget https://snort.org/downloads/snort/snort-2.9.12.tar.gz
- tar -xvzf snort-2.9.12.tar.gz
- cd snort-2.9.12
- ./configure --enable-sourcefire
- make
- sudo make install

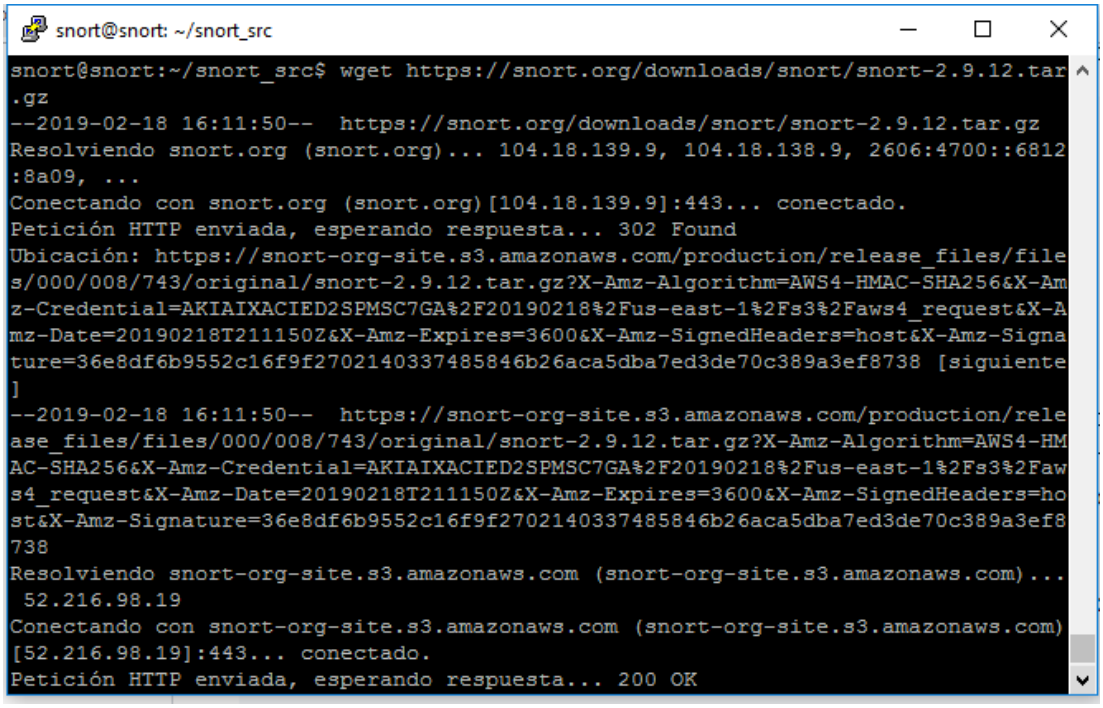


Figura 13. Descarga de snort 2.9.12.
Fuente: Elaboración propia (2018).

Una vez instalado Snort se ejecutó el siguiente comando para para actualizar las bibliotecas compartidas: sudo ldconfig y realizo un enlace simbólico al binario Snort: sudo ln -s /usr/local/bin/snort /usr/sbin/snort.

Para ver que Snort se ha instalado correctamente se usó el comando snort –v. Tal como nos muestra en la Figura 14.

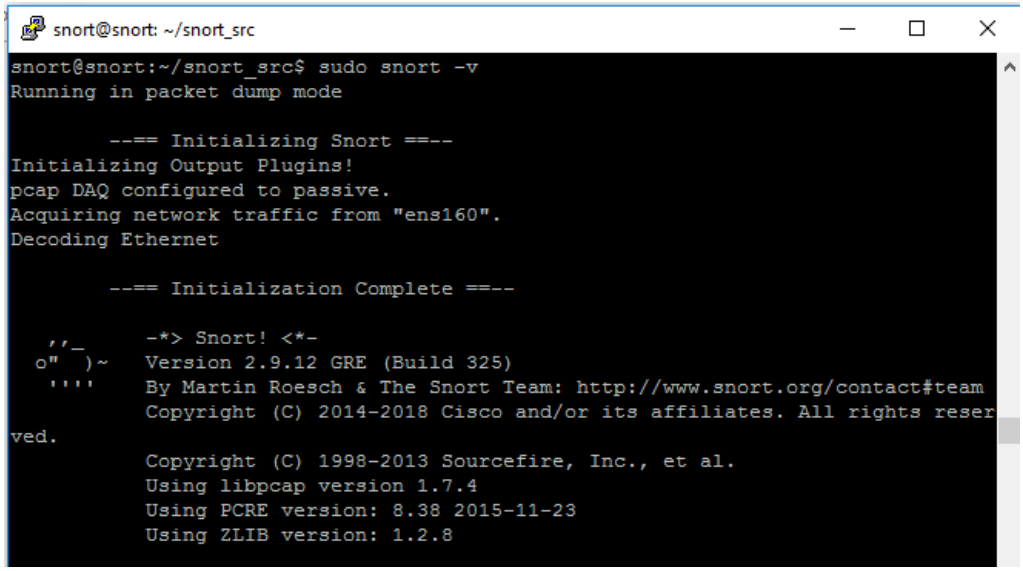


Figura 14. Verificar instalación de Snort
Fuente: Elaboración propia (2018).

4.5 Configuración de snort en modo NIDS

Se creó un usuario y un grupo con el nombre de snort para posteriormente ejecutarlo como un demonio, también se creó una serie de directorios y archivos necesarios para Snort y se estableció los permisos para dichos archivos. Se introdujeron los siguientes comandos:

- # Creando el usuario y grupo snort
- sudo groupadd snort
- sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort
- # Creando los directorios de snort:
- sudo mkdir /etc/snort
- sudo mkdir /etc/snort/rules
- sudo mkdir /etc/snort/rules/iplists
- sudo mkdir /etc/snort/preproc_rules
- sudo mkdir /usr/local/lib/snort_dynamicrules
- sudo mkdir /etc/snort/so_rules
- # Creando los archivos de Snort
- sudo touch /etc/snort/rules/iplists/black_list.rules
- sudo touch /etc/snort/rules/iplists/white_list.rules
- sudo touch /etc/snort/rules/local.rules
- sudo touch /etc/snort/sid-msg.map
- # Creando directorio de registros de snort:
- sudo mkdir /var/log/snort
- sudo mkdir /var/log/snort/archived_logs
- # Dando Permisos:
- sudo chmod -R 5775 /etc/snort
- sudo chmod -R 5775 /var/log/snort
- sudo chmod -R 5775 /var/log/snort/archived_logs
- sudo chmod -R 5775 /etc/snort/so_rules
- sudo chmod -R 5775 /usr/local/lib/snort_dynamicrules

También cambiamos la propiedad de los archivos que se crearon anteriormente:

- # Cambiando propiedad de archivos:

- sudo chown -R snort:snort /etc/snort
- sudo chown -R snort:snort /var/log/snort
- sudo chown -R snort:snort /usr/local/lib/snort_dynamicrules

Snort hace uso de algunos archivos de configuración y los preprocesadores dinámicos se copian a la fuente de snort el cual está ubicado en /etc/snort entonces se dispuso a copiar dichos archivos de configuración con los siguientes comandos:

cd ~/snort_src/snort-2.9.8.2/etc/

- sudo cp *.conf* /etc/snort
- sudo cp *.map /etc/snort
- sudo cp *.dtd /etc/snort
- cd~/snort_src/snort-2.9.8.2/src/dynamic-preprocessors/build/usr/local/lib/snort_dynamicpreprocessor/
- sudo cp * /usr/local/lib/snort_dynamicpreprocessor/

Una vez terminada toda nuestra configuración de directorios y archivos el listado de directorios de Snort vemos que quedo de la siguiente manera, Figura 15.

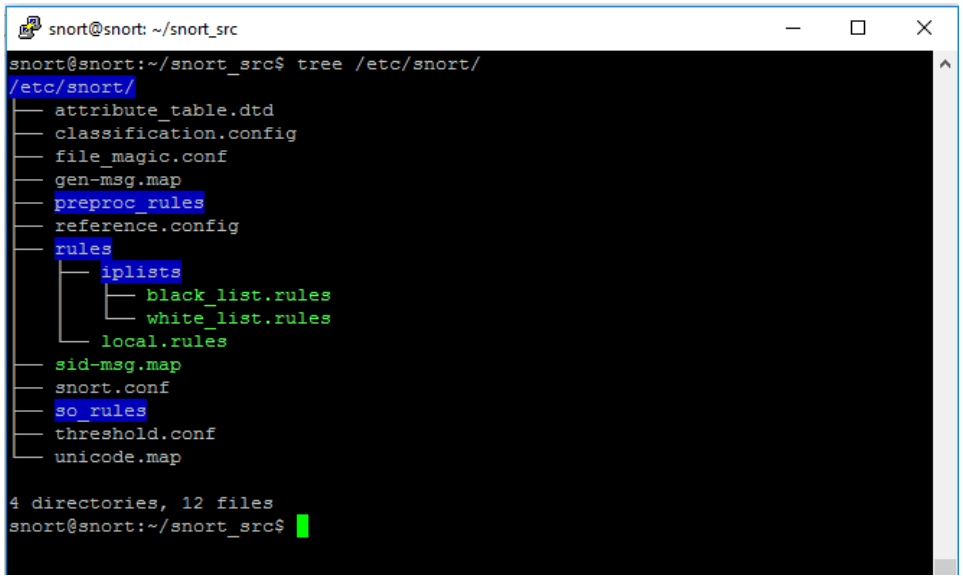


Figura 15. Listado de directorios.
Fuente: Elaboración propia (2018).

Al ya tener instalado Snort y el cual lleva consigo las reglas que se instalan por defecto, se optó por comentar dichas reglas ya que se uso PulledPork para administrar

nuestro conjunto de reglas, editando el archivo principal snort.conf con el siguiente comando: Sudo sed -i "s/include \$RULE_PATH/#include \$RULE_PATH/" /etc/snort/snort.conf

Al realizar este comando lo que se hizo fue comentar todas las palabras que conllevan "s/include \$RULE_PATH/#include \$RULE_PATH/" el cual son las reglas instaladas por defecto. Se cambió manualmente algunas configuraciones en el archivo principal snort.conf. Cambiando la siguiente línea ipvar HOME_NET any. Por ipvar HOME_NET 192.168.96.0/24 que es nuestra red principal. Tal como nos muestra la Figura 16.

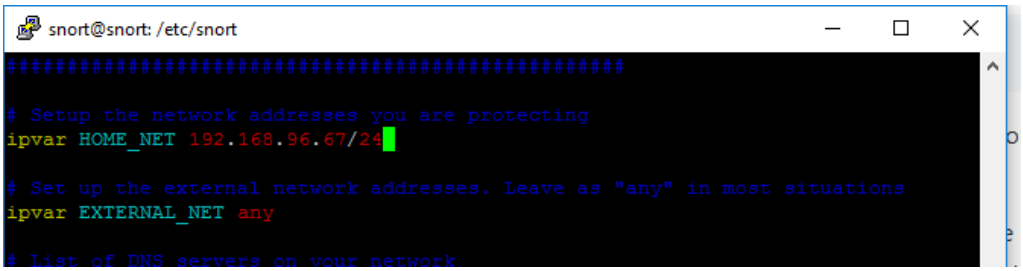


Figura 16. Insertando Rango de IP.
Fuente: Elaboración propia (2018).

Ya que anteriormente se creó los archivos de configuración entonces se editó las líneas 104 en adelante para el re direccionamiento de nuestras reglas. Como nos muestra en la Figura 17.

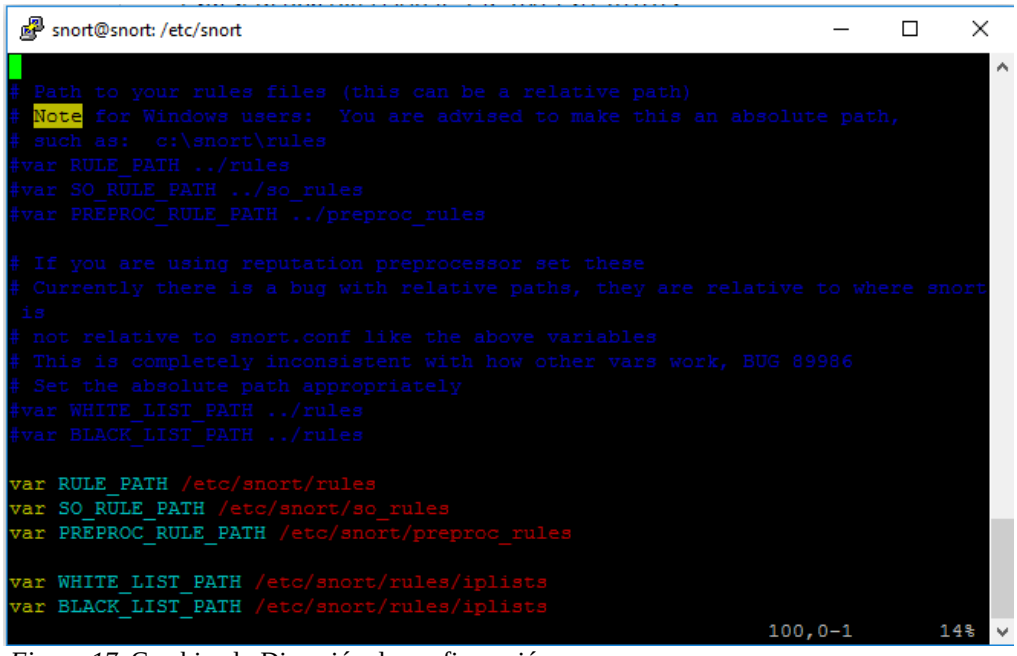


Figura 17. Cambiando Dirección de configuración.
Fuente: Elaboración propia (2018).

También habilitamos la línea include $RULE_PATH/local.rules para facilitar las pruebas de Snort. Figura 18.

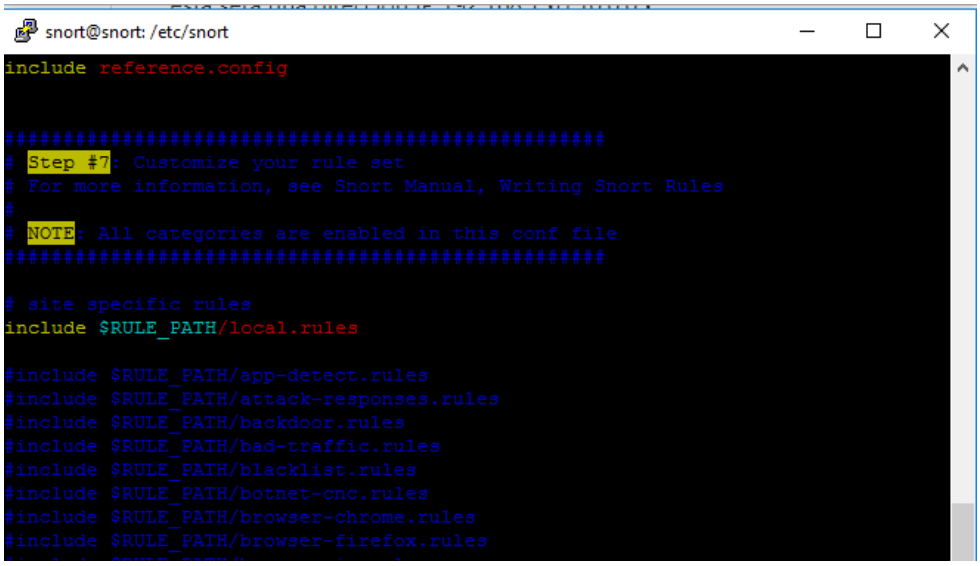


Figura 18. Habilitando LocalRules.
Fuente: Elaboración propia (2018).

Una vez que ya terminamos de configurar todo se usamos el siguiente comando para verificar que nuestro archivo snort.conf está configurado correctamente: sudo snort -T -i ens160 -c /etc/snort/snort.conf para ver que todos los archivos necesarios estén correctos. Figura 19.

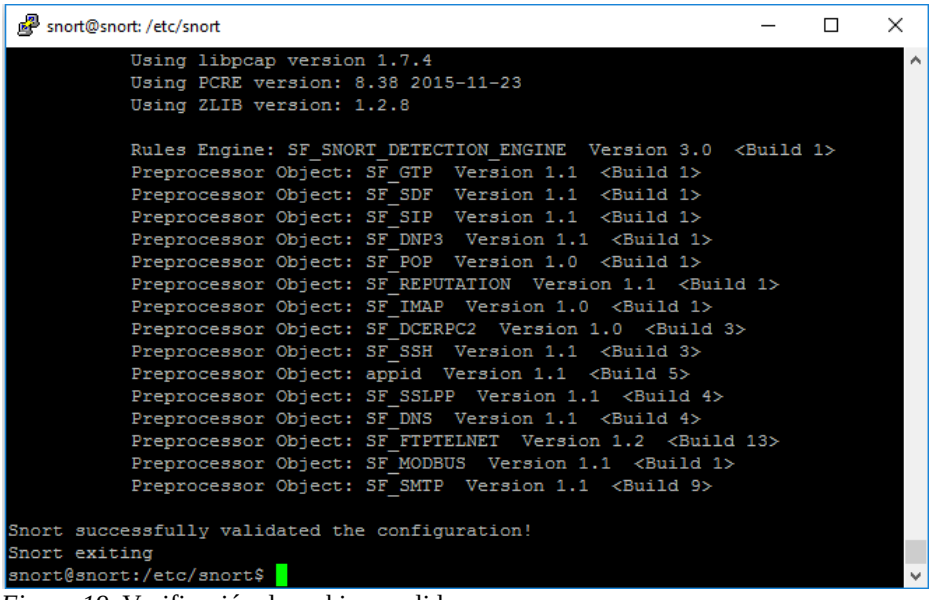


Figura 19. Verificación de archivos validos.
Fuente: Elaboración propia (2018).

Para ver que que snort ya detecta intrusos escribimos una simple regla, se editó el archivo /etc/snort/rules/local.rules y se definió una regla. Figura 20.

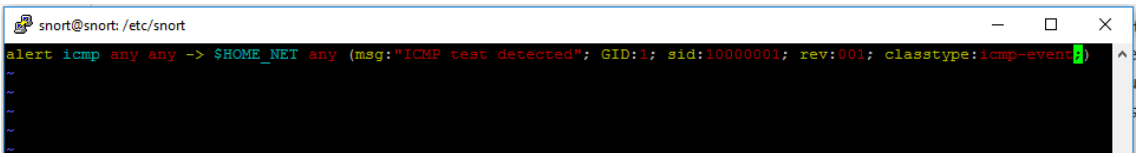


Figura 20. Regla de Ping desde otra máquina hacia Snort.
Fuente: Elaboración propia (2018).

Como se trabajó con Barnyard2 y Barnyard2 no lee la meta información de las alertas del archivo local.rules, entonces al agregar nuevas reglas generara errores con PulledPork entonces se configuro el archivo /etc/snort/sid-msg.map para que pueda agregar las reglas correctamente. Figura 21.

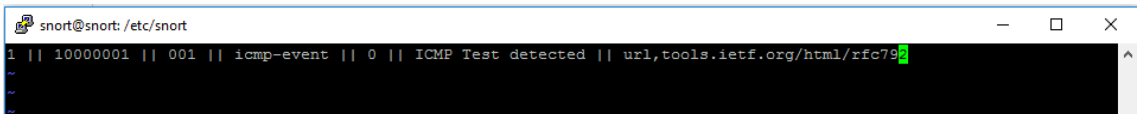


Figura 21. Indicar a Barnyard2 que la regla inicia en 10000001.
Fuente: Elaboración propia (2018).

Como se hizo cambios en la configuración de snort se probó nuevamente la configuración de snort con el siguiente comando:

sudo snort -T -c /etc/snort/snort.conf -i ens160

Y vemos satisfactoriamente que la regla que creamos se ejecutó correctamente. Figura 22.

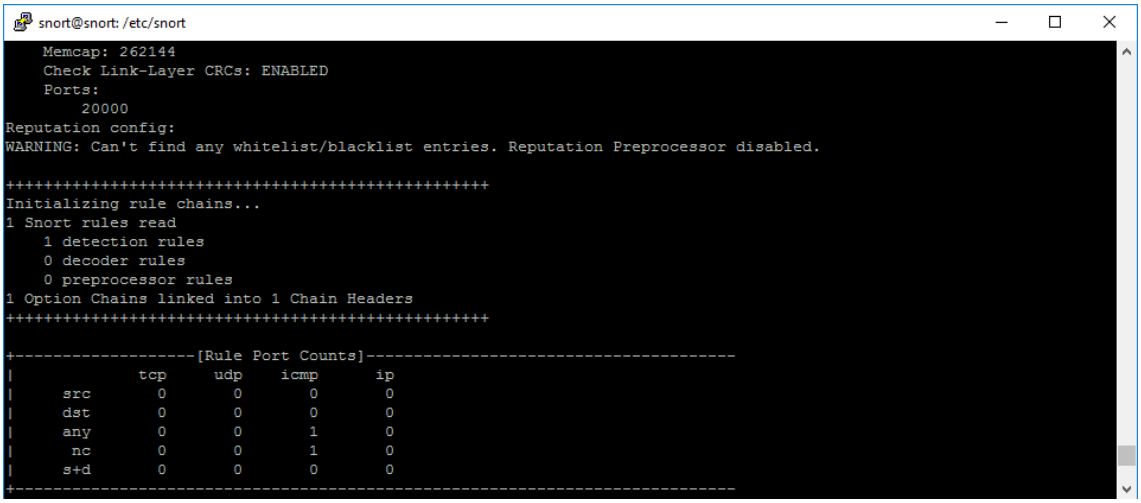


Figura 22. Ver la ejecución de la regla.
Fuente: Elaboración propia (2018).

Ahora para probar que Snort en verdad hizo su trabajo se ejecutó snort con el siguiente comando: Figura 23. sudo /usr/local/bin/snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i ens160

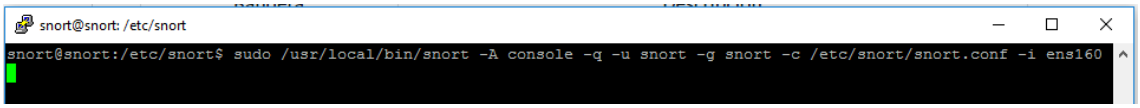


Figura 23. Ejecutando Snort.
Fuente: Elaboración propia (2018).

Vemos que al ejecutar Snort no nos mostró nada, sin embargo, al realizar un ping a nuestro servidor desde otra máquina vimos que Snort si analiza los paquetes según la regla previamente puesta. Figura 24.

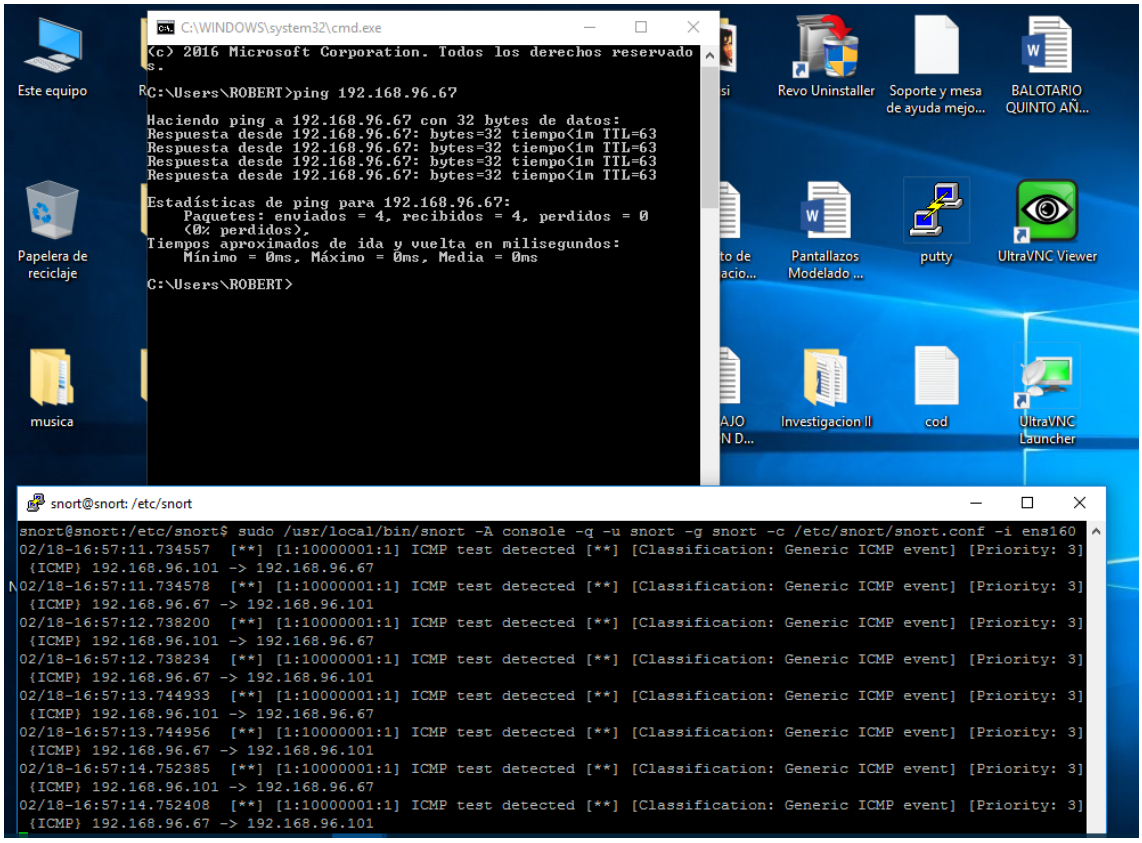


Figura 24. Prueba funcionamiento snort desde otra maquina.
Fuente: Elaboración propia (2018).

4.6 Configuración de snort en modo IPS

Para poner snort en modo IPS debemos de volver a compilar el DAQ la razón es porque al instalar en modo NIDS no instala el paquete NFQ que es necesario para que snort pueda funcionar en modo IPS. Instalaremos esta librerías adicionales sudo apt-get install -y

libnghttp2-dev para el funcionamiento de IPS también instalaremos el paquete faltante de NFQ, con el siguiente comando sudo apt-get install libnetfilter-queue-dev, nos dirigimos al directorio de daq y volvemos a ejecutar el comando sudo ./configure ahora veremos que ya tenemos instalado el paquete NFQ. Figura 25.

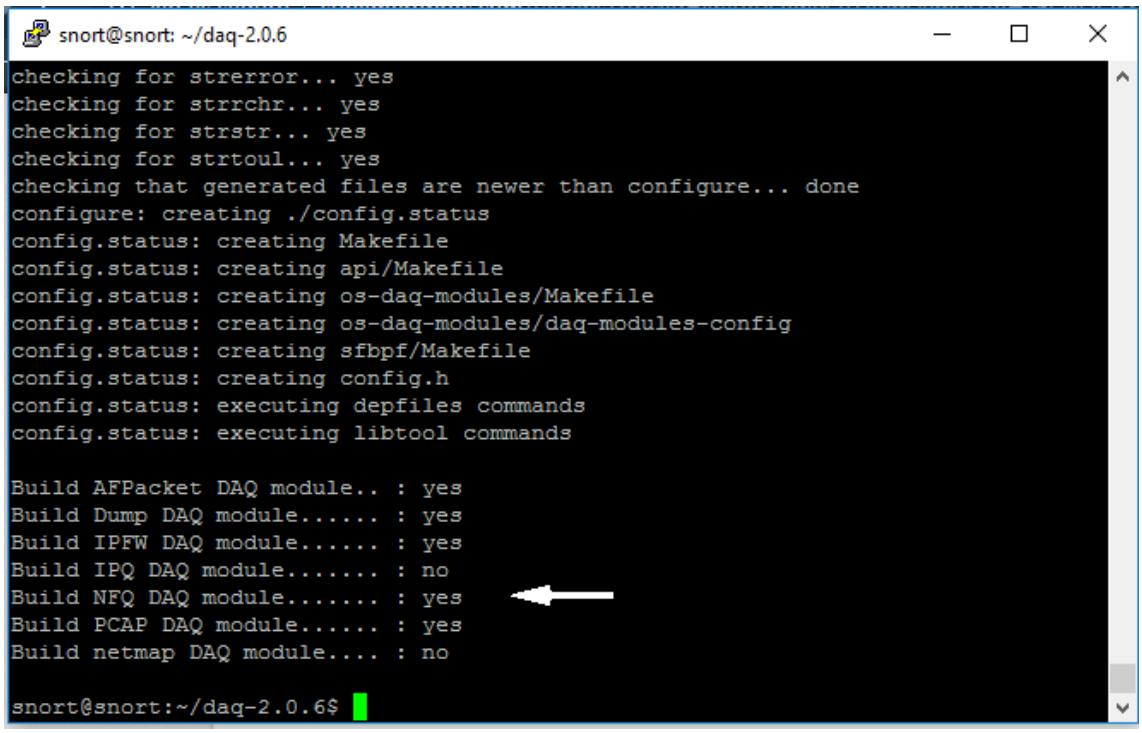


Figura 25. Instalando NFQ.
Fuente: Elaboración propia (2018).

Ahora pasamos a editar el archivo snort.conf para insertar valores de configuración IPS. Ingresamos los siguientes valores. Figura 26.

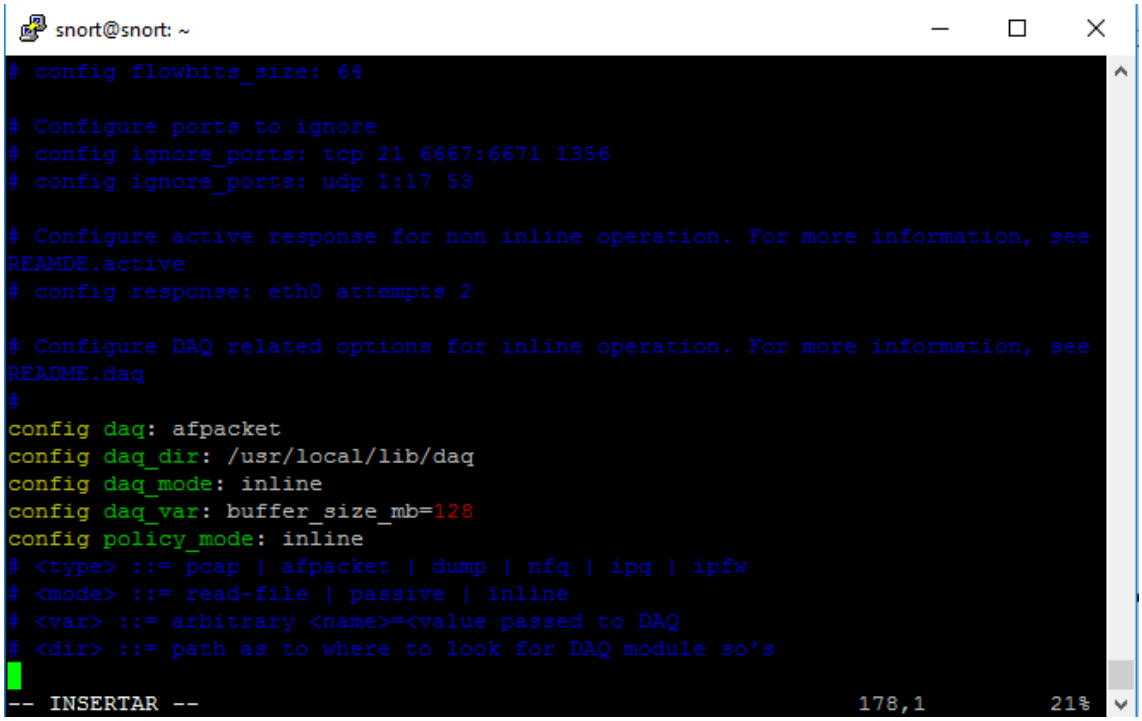


Figura 26. Configuración modo IPS.
Fuente: Elaboración propia (2018).

Como estamos trabajando con afpacket en la configuración de snort para iniciar snort se ejecuta mencionando 2 interfaces graficas a escuchar como en el siguiente comando:

sudo /usr/local/bin/snort -A console -Q -c /etc/snort/snort.conf -i ens160:ens224 -N

4.7 Optimización de recursos instalando Barnyard2

El IDS snort requiere de muchos recursos para realizar los eventos para que el usuario pueda ver, ya que al realizar el monitoreo del tráfico de la red snort almacena toda esa información en archivos log entonces para optimizar estas operaciones haremos uso de Barnyard2 para que pueda almacenar toda esa información en una base de datos MYSQL para de esa manera poder ver, buscar y perfilar los eventos, entonces ahora configuraremos snort para que genere eventos en formato binario en un solo lugar para que de esa manera Barnyard2 lea los eventos y los guarde en nuestra base de datos MySQL. Requisitos que se instaló para la utilización de Barnyard2. Figura 27.

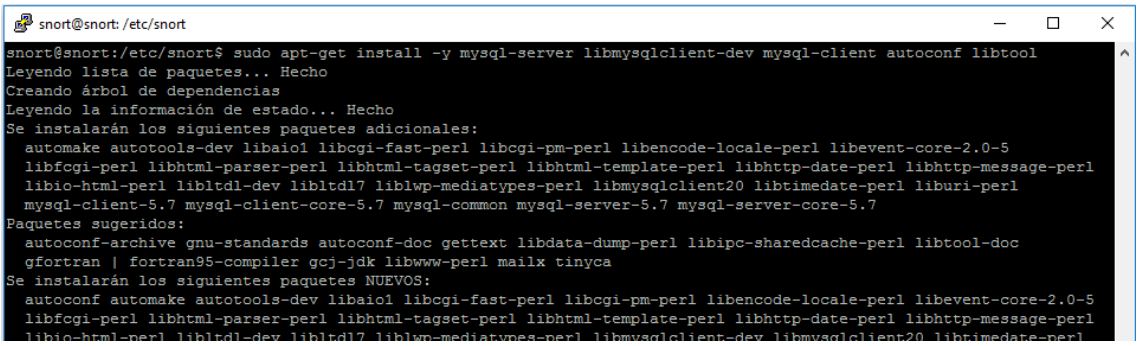


Figura 27. Requisitos para instalar Barnyard2.
Fuente: Elaboración propia (2018).

Una vez se instaló los requisitos para Barnyard2 se indicó a Snort que debe permitir emitir sus alertas en formato binario para que Barnyard2 pueda procesarlo se edito el archivo /etc/snort/snort.conf y se introdujo la siguiente línea: output unified2: filename snort.u2, limit 128. Figura 28.

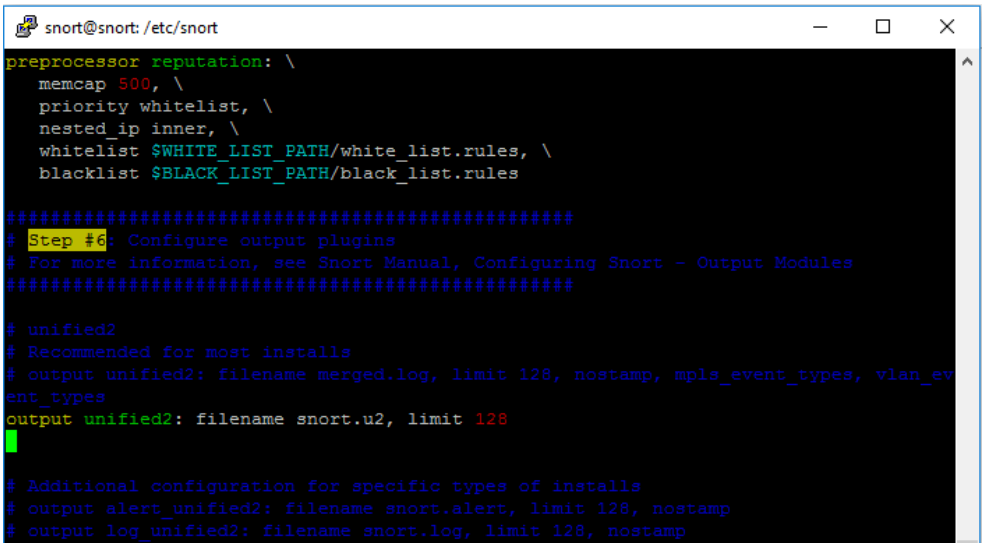


Figura 28. Comando para leer las alertas en formato binario.
Fuente: Elaboración propia (2018).

Una vez configurado todo se procedió a la descarga e instalación de Barnyard2. Figura 29.

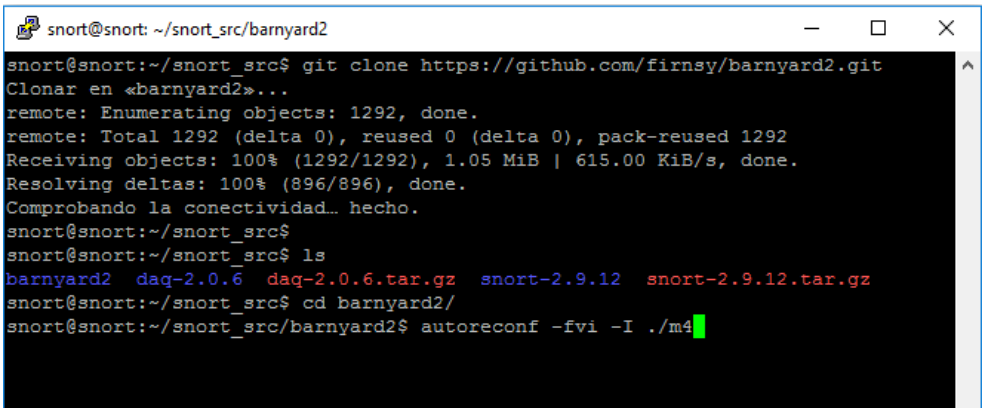


Figura 29. Descargando e instalando Barnyard2.
Fuente: Elaboración propia (2018).

Barnyard2 necesita hacer uso de la biblioteca dnet.h asi que se creó un enlace flexible de dnet.h a dubdnet.h para evitar problemas se utilizó el siguiente comando:

sudo ln -s /usr/include/dumbnet.h /usr/include/dnet.h

sudo ldconfig

Una vez realizado eso se indicó la instalación de la biblioteca MySql con el siguiente comando:

./configure --with-mysql --with-mysql-libraries=/usr/lib/x86_64-linux-gnu

Y se continúo con la instalación:

- Make
- Make install

Cuando Barnyard2 ya estuvo instalado se realizó la copia y la creación de algunos archivos que Barnyard2 requiere para ejecutarse. Figura 30.

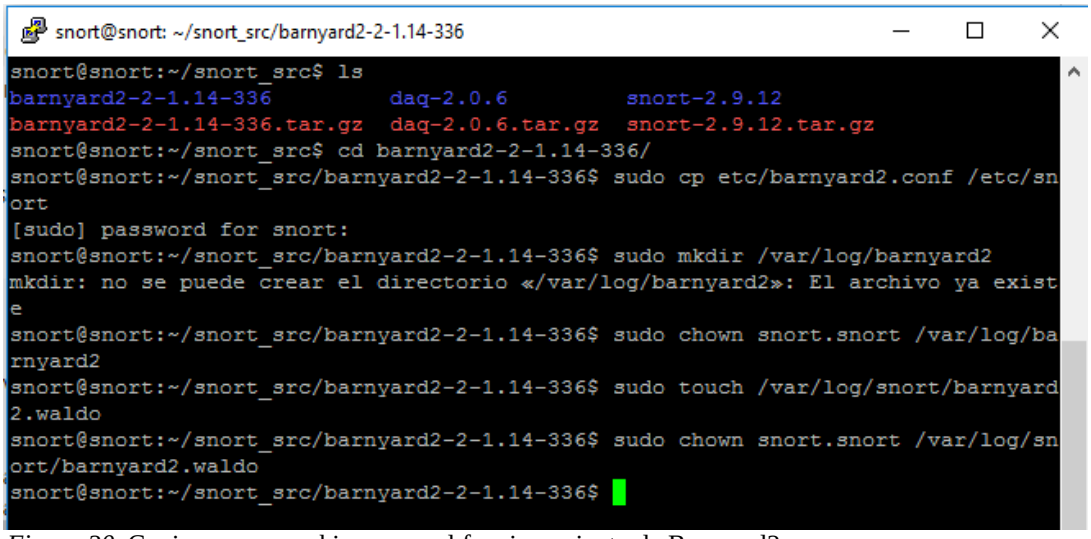


Figura 30. Copiar y crear archivos para el funcionamiento de Barnyard2.
Fuente: Elaboración propia (2018).

Ya que Barnyard2 guarda nuestras alertas en nuestra base de datos MySql se procedió a crear la base de datos snort, generar tablas y crear usuario. Figura 31,32 y 33.

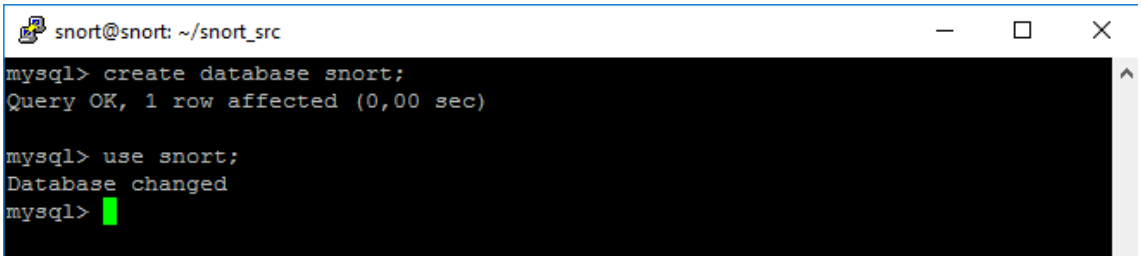


Figura 31. Creación de Base de datos.
Fuente: Elaboración propia (2018).

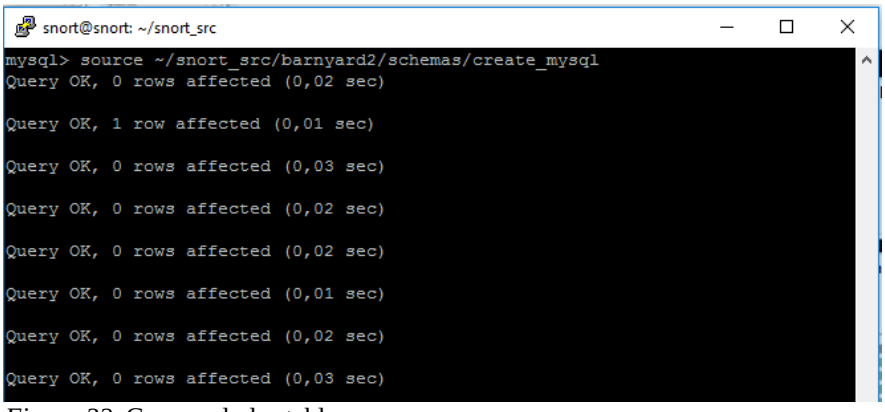


Figura 32. Generando las tablas.
Fuente: Elaboración propia (2018).

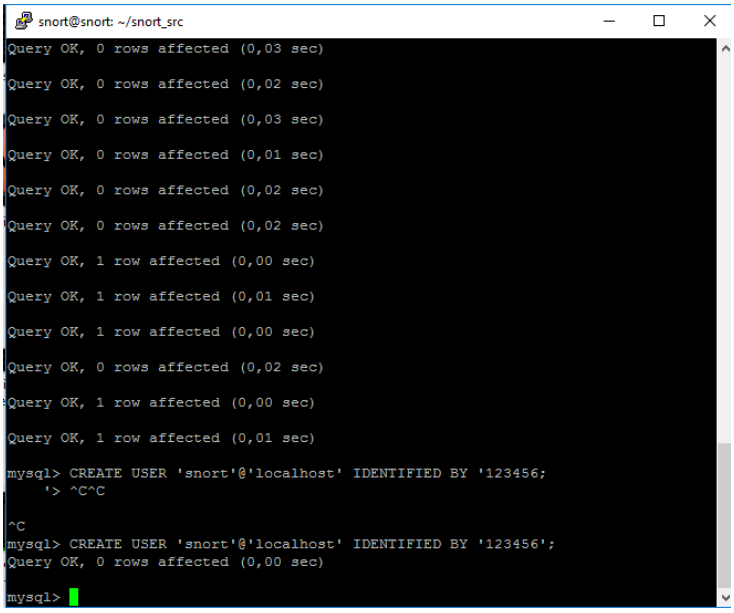


Figura 33. Creando usuario y contraseña.
Fuente: Elaboración propia (2018).

Ahora para que Barnayard2 pueda conectarse a la base de datos MySql se edito el archivo /etc/snort/barnyard2.conf y se introdujo al final del archivo la siguiente línea:

output database: log, mysql, user=snort password=123456 dbname=snort host=localhost

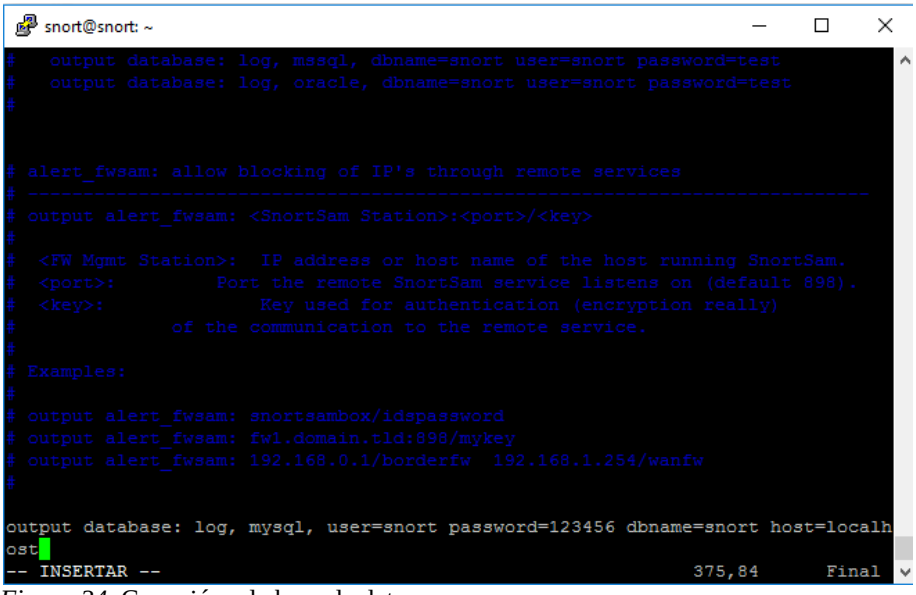


Figura 34. Conexión a la base de datos.
Fuente: Elaboración propia (2018).

Dado que la contraseña se almacena en texto sin cifrar en el barnyard2.confarchivo, se ejecutó el siguiente comando: sudo chmod o-r /etc/snort/barnyard2.conf

Para ver que Barnyard2 se instaló y configuro correctamente se introdujo el siguiente comando: Figura 35.

sudo barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -w /var/log/snort/barnyard2.waldo -g snort -u snort

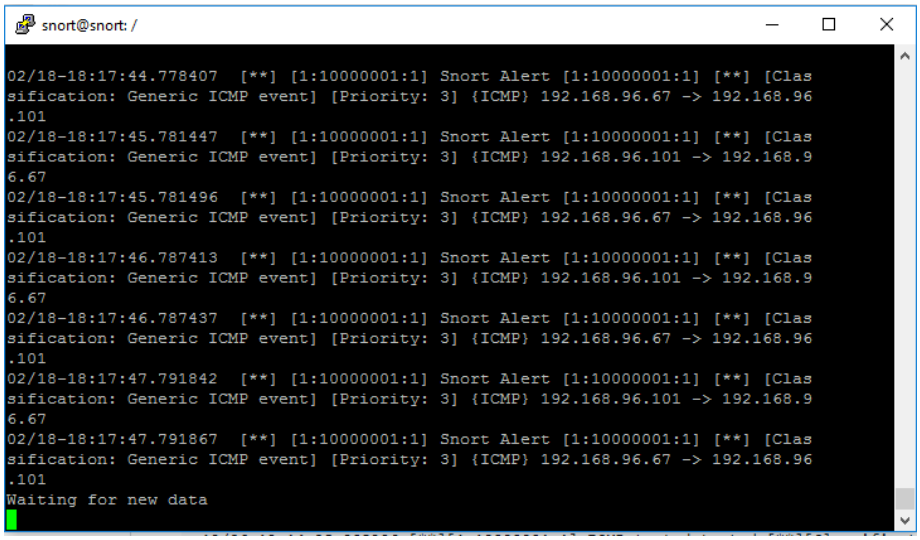


Figura 35. Haciendo Correr Barnyard2
Fuente: Elaboración propia (2018).

Para verificar los eventos en la base de datos y ver que Barnyard2 en verdad escribió los eventos se introdujo el siguiente comando: mysql -u snort -p -D snort -e "select count(*) from event".

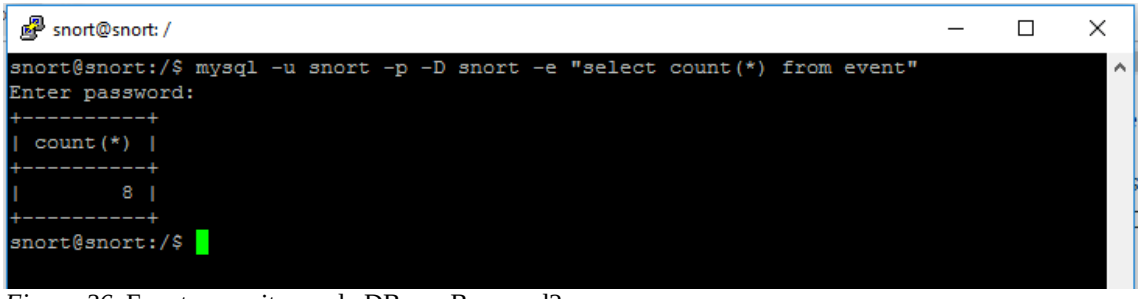


Figura 36. Eventos escritos en la DB por Barnyard2.
Fuente: Elaboración propia (2018).

4.8 Actualización de reglas

Utilizaremos las reglas que nos brindan al tener una cuenta en snort.org y haremos uso de nuestro Oinkcode.

Descargamos nuestras reglas con nuestro respectivo Oinkcode y editaremos el archivo pulledpork.conf que se encuentra en sudo nano /etc/snort/pulledpork.conf. Como nos muestra en la figura 37.

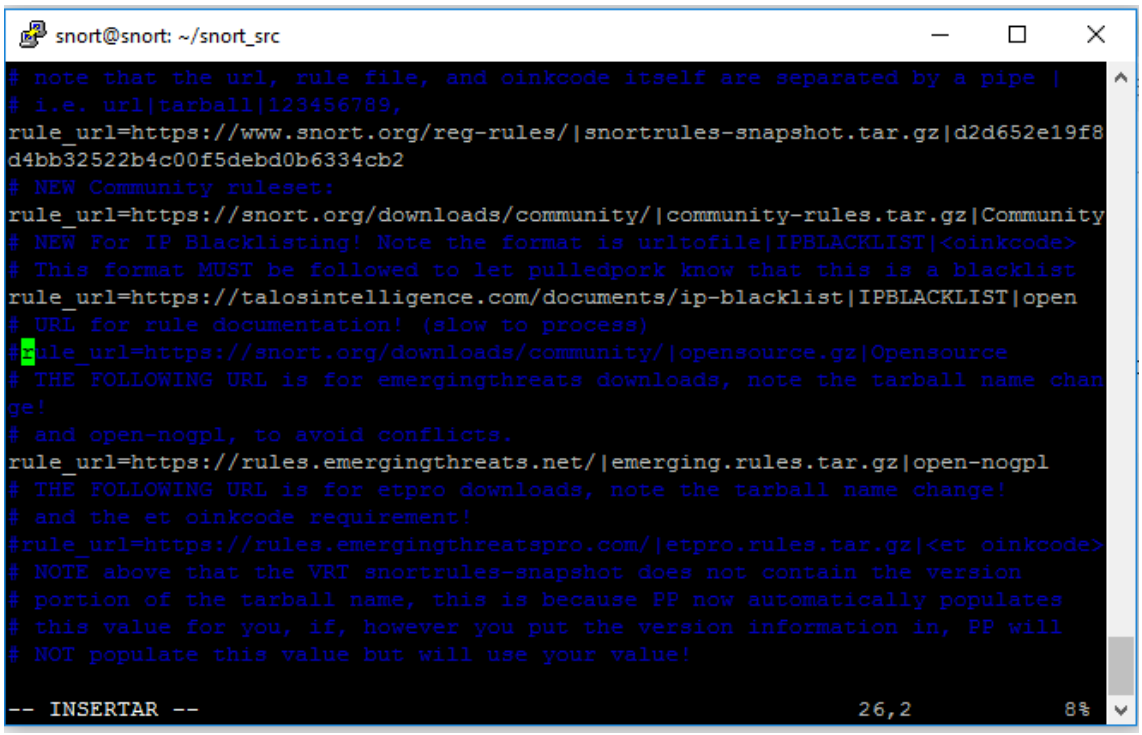


Figura 37. Configuración del archivo pulledpork.conf.
Fuente: Elaboración propia (2018).

Al realizar toda la configuracion y ejecutar el siguiente comando:

sudo /usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf –l

Veremos que ya tenemos más de 60000 reglas descargadas el cual daremos uso para monitorear todo el tráfico de la red de la Universidad Peruana Unión – FJ. Figura 38.

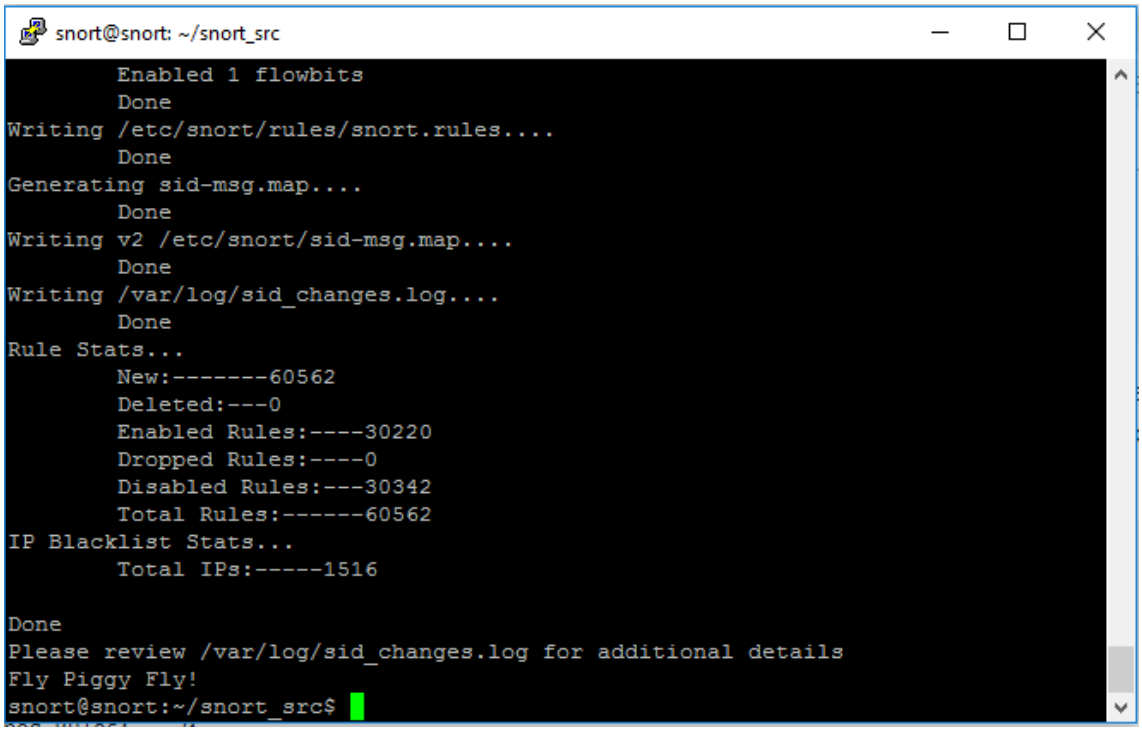


Figura 38. Reglas descargadas con el oinkcode.
Fuente: Elaboración propia

Ahora que ya descargo satisfactoriamente las reglas se debio haber creado un archivo llamado snort.rules dentro de la carpeta /etc/snort/rules/snort.rules es en ese archivo donde se guardaron todas las reglas descargadas, ya que ahora tenemos un nuevo archivo debemos editar nuestra configuración de snort y decirle que use estas reglas, realizamos esto abriendo el archivo de configuración de snort /etc/snort/snort.conf. Figura 39.

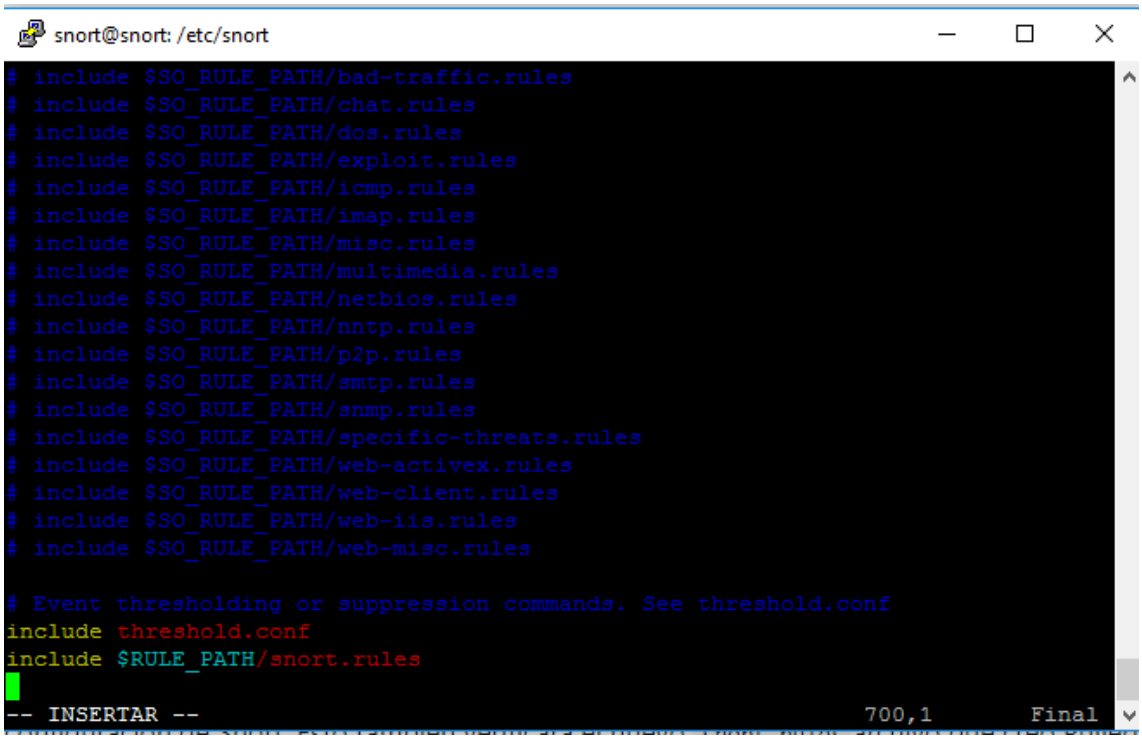


Figura 39. Incluyendo el archivo de reglas en snort.conf.
Fuente: Elaboración propia (2018).

Ahora usaremos el comando sudo snort –T –c /etc/snort/snort.conf –i ens160. Para verificar que nuestra configuración de snort esta correcta. Una vez terminada esta configuración realizaremos un Crontab para que se ejecute todos los días a las 4 am con 1 min. Figura 40.

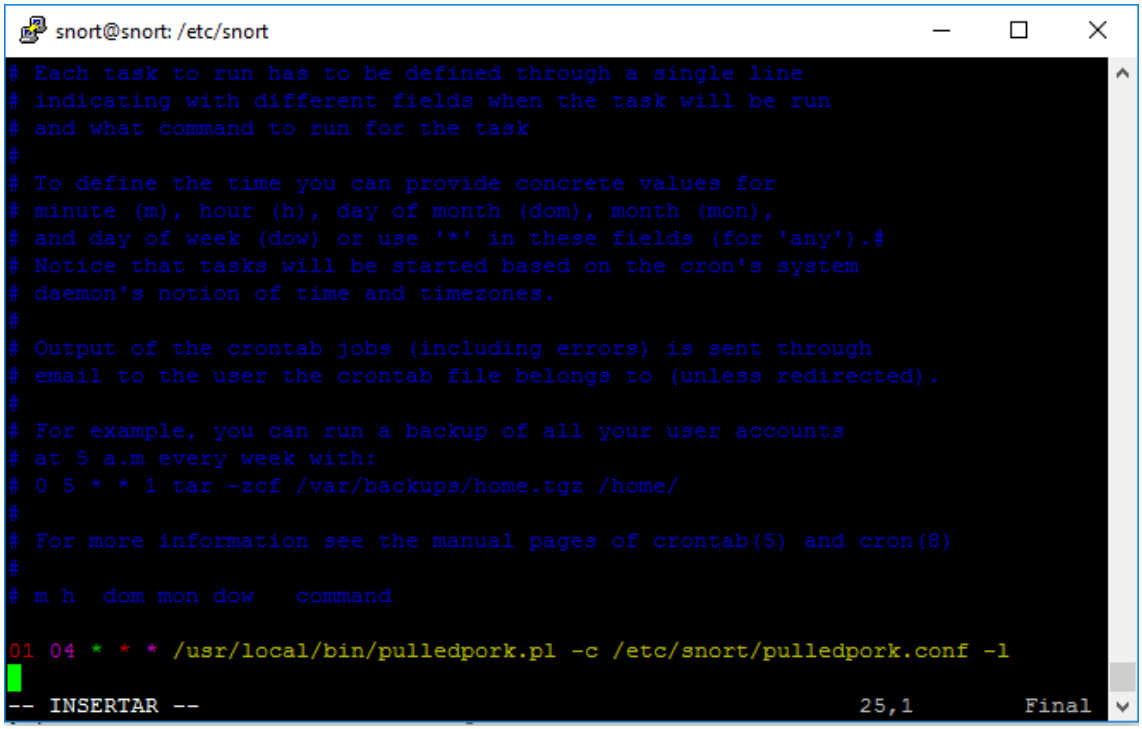


Figura 40. Configurando crontab para pulledpork.
Fuente: Elaboración propia (2018).

4.9 Configuración de los archivos daemons

Ahora ya tenemos todo configurado sin embargo los sistemas no se están iniciando solos, entonces crearemos los demonios para snort y para barnyard2 para que puedan iniciar junto con el sistema operativo. Crearemos un archivo para el inicio de snort en la siguiente dirección, sudo vim /lib/systemd/system/snort.service estando dentro de este archivo pondremos lo siguiente: Figura 41.

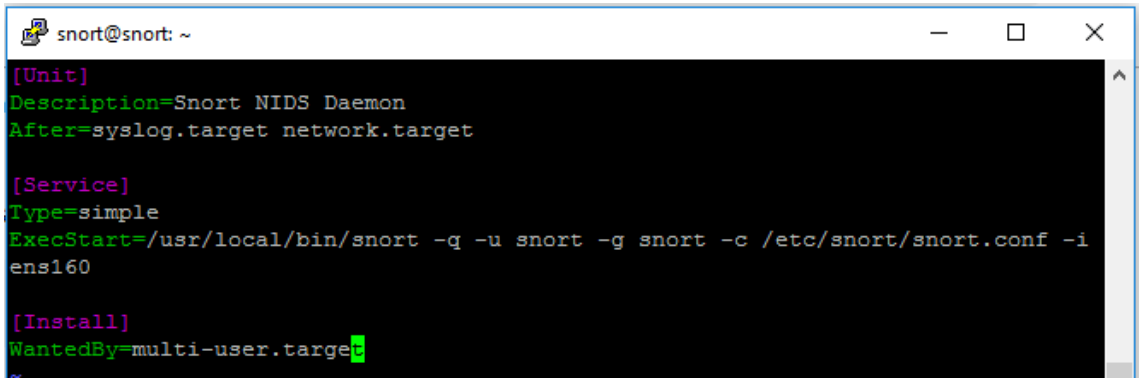


Figura 41. Archivo damm de snort.
Fuente: Elaboración propia (2018).

Terminado de crear ejecutamos los siguientes comandos para decirle al systemD que inicie en el arranque del sistema.

sudo systemctl enable snort

sudo systemctl start snort

systemctl status snort

Ahora pasamos a crear el archivo demonio para barnyard2 en la siguiente dirección, sudo vim /lib/systemd/system/barnyard2.service, introduciendo lo siguiente: Figura 42

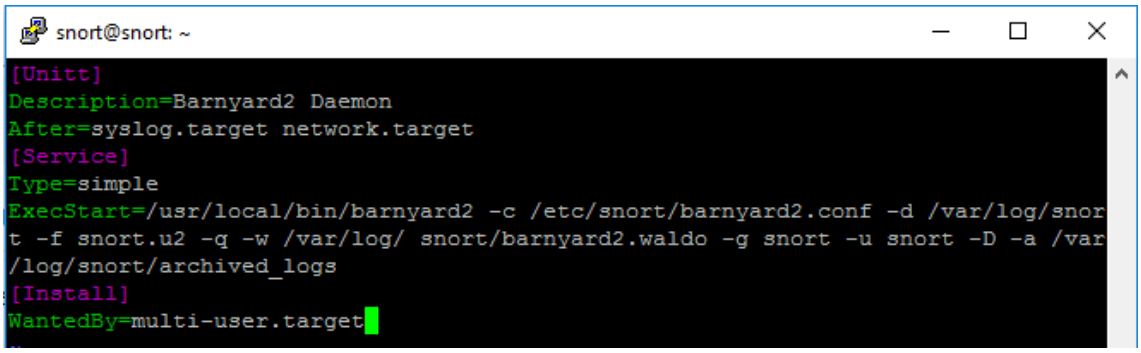


Figura 42. Archivo damm de barnyard2.
Fuente: Elaboración propia (2018).

De igual manera configuramos este archivo para que inicie en el arranque del sistema.

sudo systemctl enable barnyard2

sudo systemctl start barnyard2

systemctl status barnyard2

4.10 Configuración de la interfaz gráfica para snort

Para poder tener una mejor visualización del monitoreo del tráfico de la red se optó por configurar una interfaz web para snort. Para esta investigación se usó el GUI snorby porque es una interfaz web un poco más amigable que los otros GUIs.

Para hacer uso de snorby se requieren ciertas librerías como requisito previo a la instalación. Las librerías requeridas son las siguientes. Figura 43.

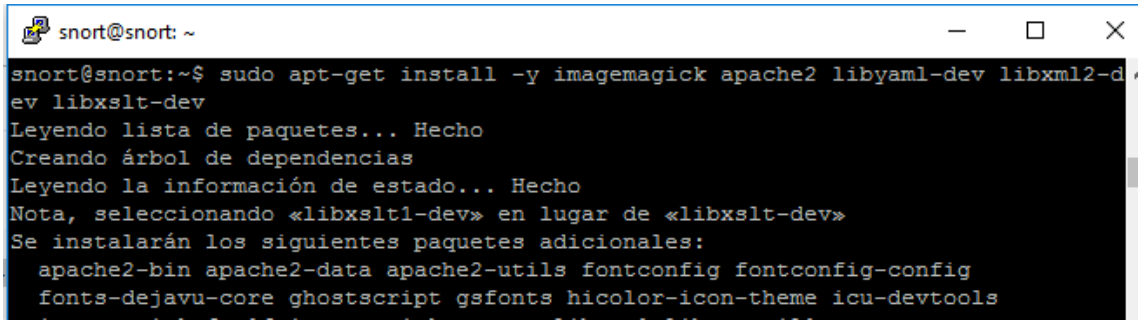


Figura 43. Librerias necesarias para snorby.
Fuente: Elaboración propia (2018).

Snorby utiliza Ruby on Rays para proporcionar una interfaz web, instalaremos ruby en una versión específica para eso necesitamos instalar el repositorio Brightbox.

sudo apt-add-repository ppa:brightbox/ruby-ng

Una vez instalada el repositorio de Brightbox pasamos a descargar Ruby 2.3. Figura 44

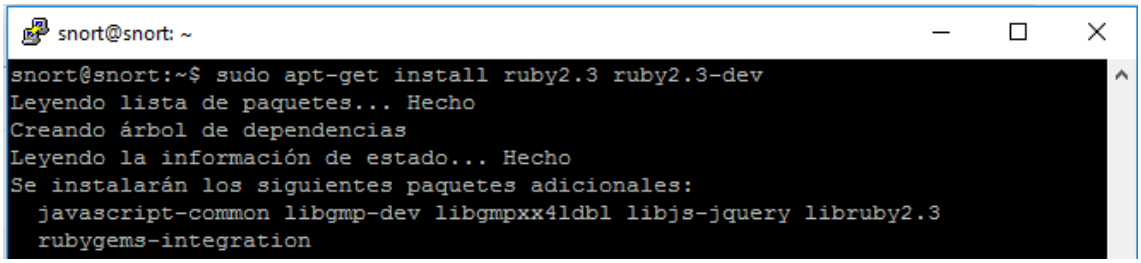


Figura 44. Instalando Ruby 2.3.
Fuente: Elaboración propia (2018).

Ya que ruby trabaja a base de gemas procederemos a descargar las gemas necesarias para la instalación de snorby. Las gemas requeridas son las siguientes:

sudo gem install wkhtmltopdf

sudo gem install bundler

sudo gem install rails

sudo gem install rake --version=0.9.2

Una vez ya instalada las gemas iremos al repositorio de Git donde esta alojado snorby y procederemos a descargar snorby git clone https://github.com/Snorby/snorby.git. Una vez descargado snorby copiaremos el directorio snorby a nuestro directorio de servidor web con

el siguiente comando sudo cp -r ./snorby/ /var/www/snorby/. Ingresamos a l directorio /var/www/snorby/. Una vez dentrol directorio ejecutaremos el comando sudo gem update –system para que reconozca las gemas instaladas, ahora ingresamos el comando bundle install para descargar las demás gemas necesarias para usar snorby. Figura 45.

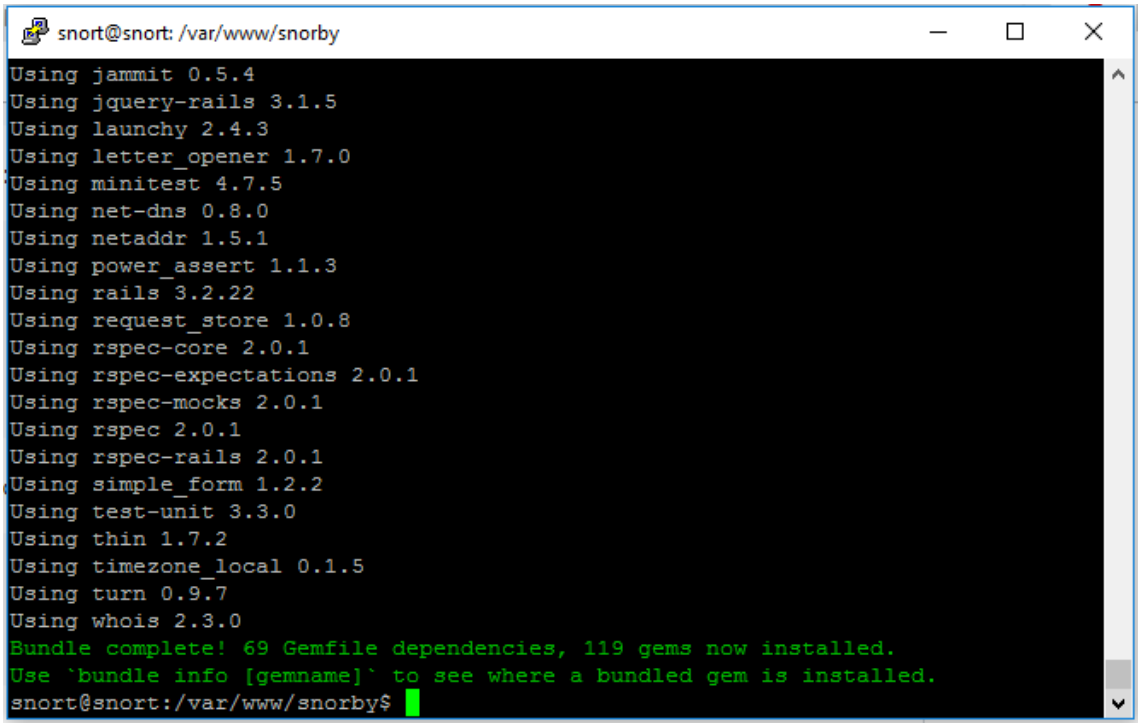


Figura 45. Gemas instaladas con bundle install
Fuente: Elaboración propia (2018).

Ahora en el mismo directorio copiaremos el archivo database.yml,example al mismo directorio para habilitar y simplemente quede database.yml. Ingresamos el siguiente comando sudo cp /var/www/snorby/config/database.yml.example /var/www/snorby/config/database.yml, ahora abrimos el archivo database.yml para ingresar el usuario y contraseña de nuestra base de datos. Asi como nos muestra la Figura 46.

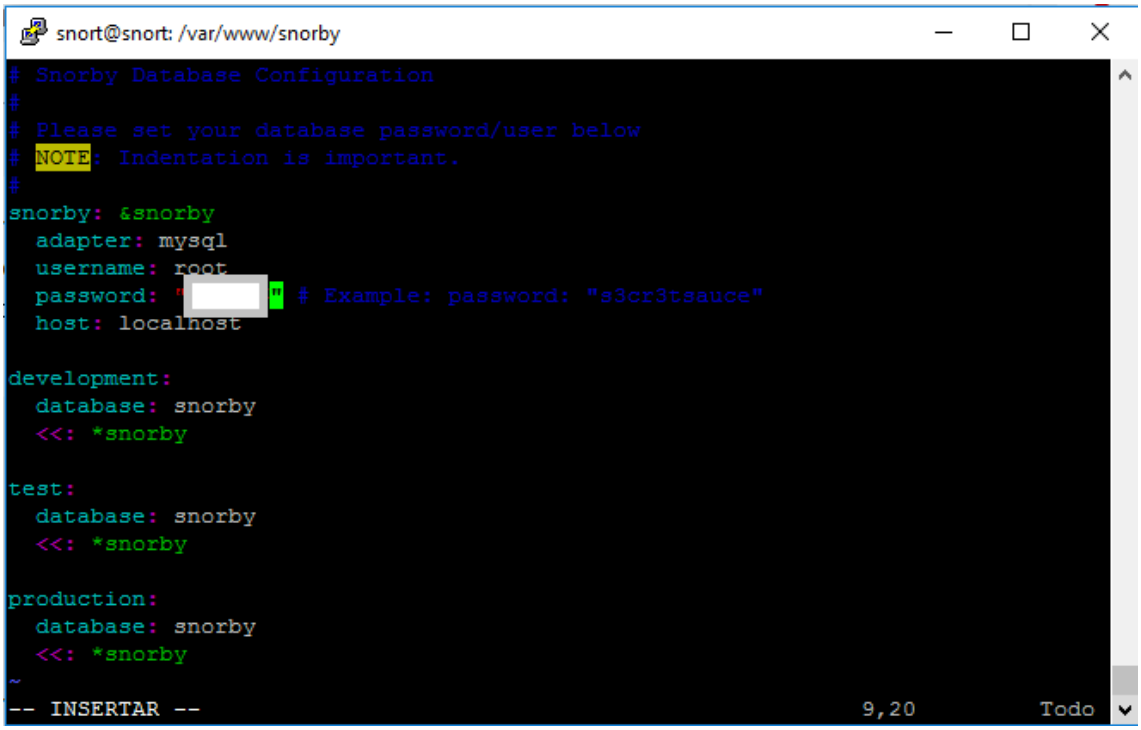


Figura 46. Configuración para conexion a la base de datos.
Fuente: Elaboración propia (2018).

Ahora crearemos el archivo de configuración de snorby y actualizarlo para que apunte a la versión correcta wkhtmlpdf con los siguientes comandos:

sudo cp /var/www/snorby/config/snorby_config.yml.example /var/www/snorby/config/snorby_config.yml

sudo sed -i s/"\/usr\/local\/bin\/wkhtmltopdf"/"\/usr\/bin\/wkhtmltopdf"/g /var/www/snorby/config/snorby_config.yml

Procedemos a instalar snorby con el siguiente comando el cual descargara algunas gemas adicionales y creara una nueva base de datos llamada snorby. Figura 47.

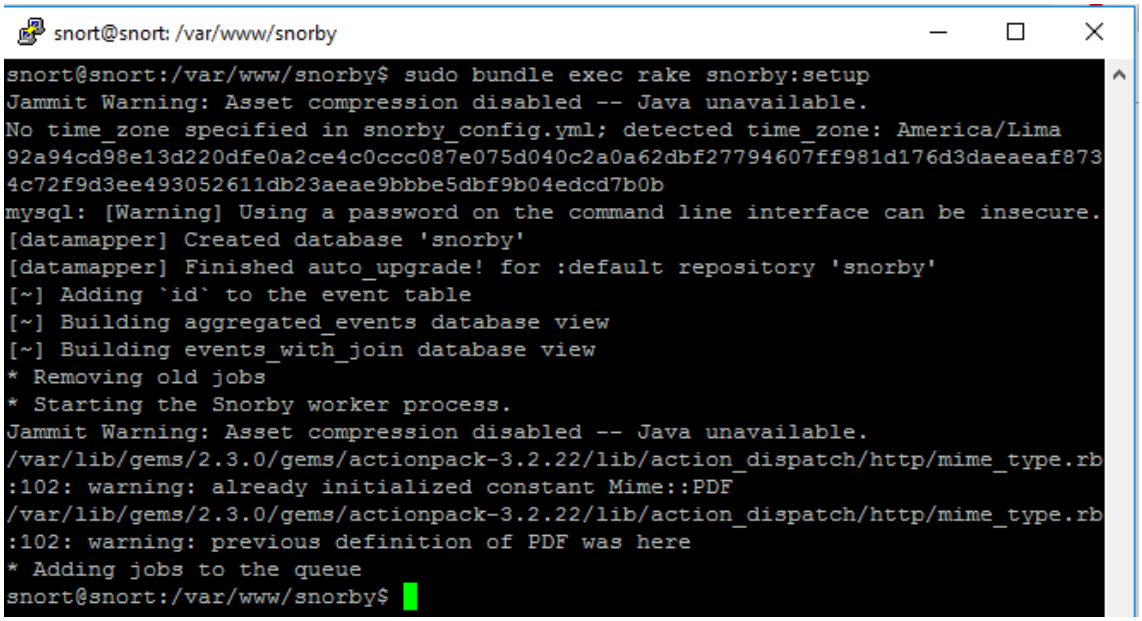


Figura 47. Instalando snorby y creacion de db.
Fuente: Elaboración propia (2018).

Ahora editamos la base de datos para dar privilegios y crear un nuevo usuario. Figura 48.

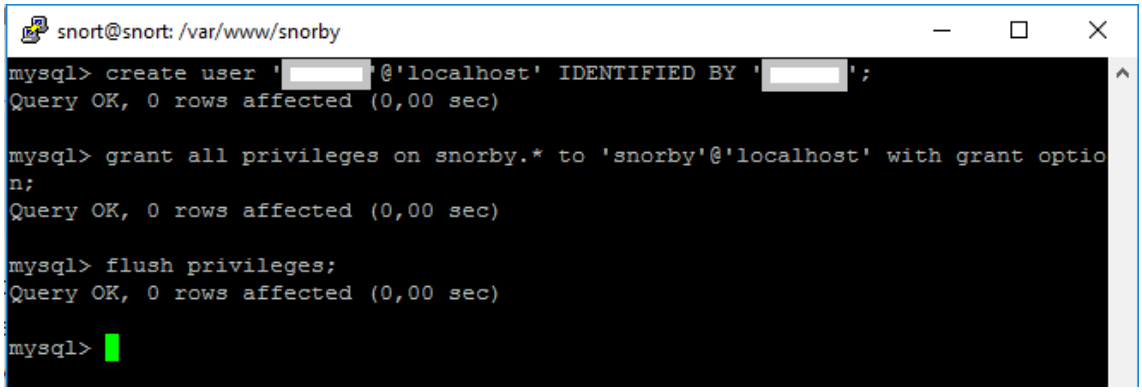


Figura 48. Crear usuario y dar privilegios de snorby.
Fuente: Elaboración propia (2018).

Ahora que tenemos un nuevo usuario en mysql para snorby editamos el archivo database.yml e introducimos nuestro usuario y contraseña. Figura 49.

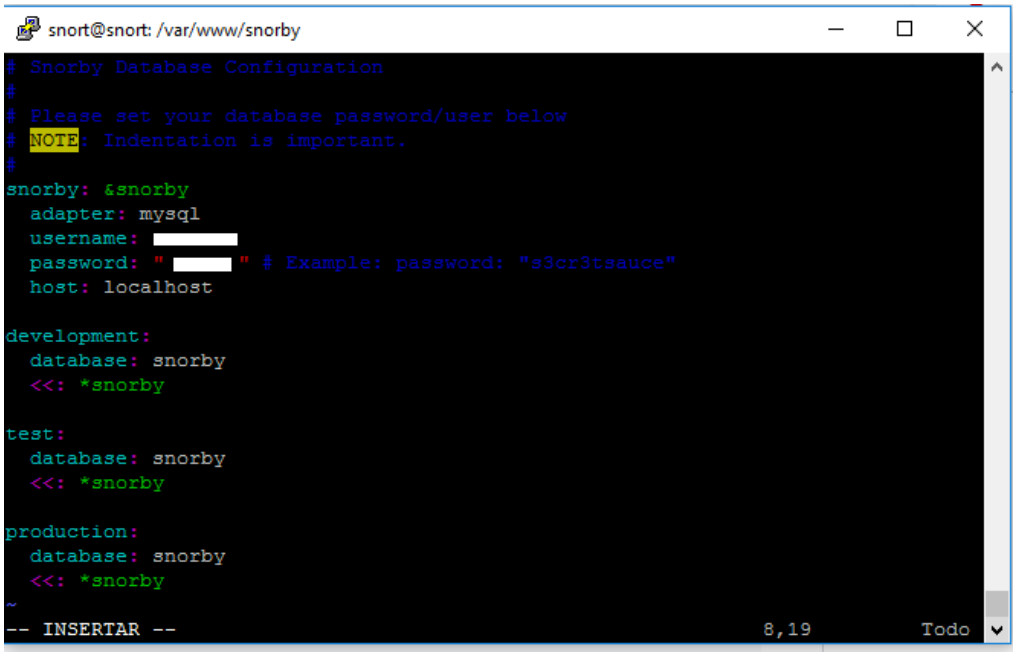


Figura 49. Configuración de conexión a base de datos.
Fuente: Elaboración propia (2018).

Ya tenemos todo configurado ahora podemos probar que snorby funcione correctamente introducimos el siguiente comando en /var/www/snorby.

sudo bundle exec rails server -e production

Vemos que snorby se inicia en el puero 3000 y ya podemos visualizar la interfaz gráfica. Figura 50 y 51.

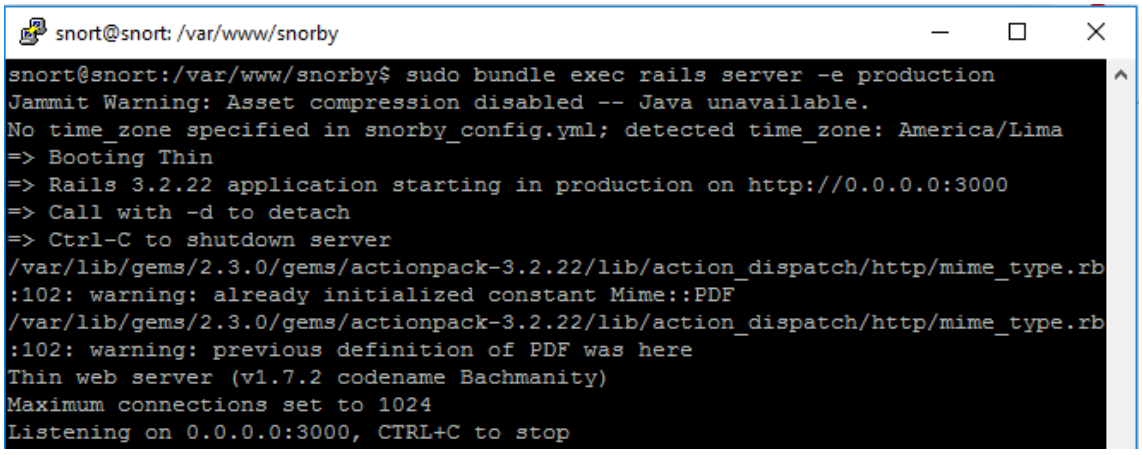


Figura 50. Iniciando snorby.
Fuente: Elaboración propia (2018).

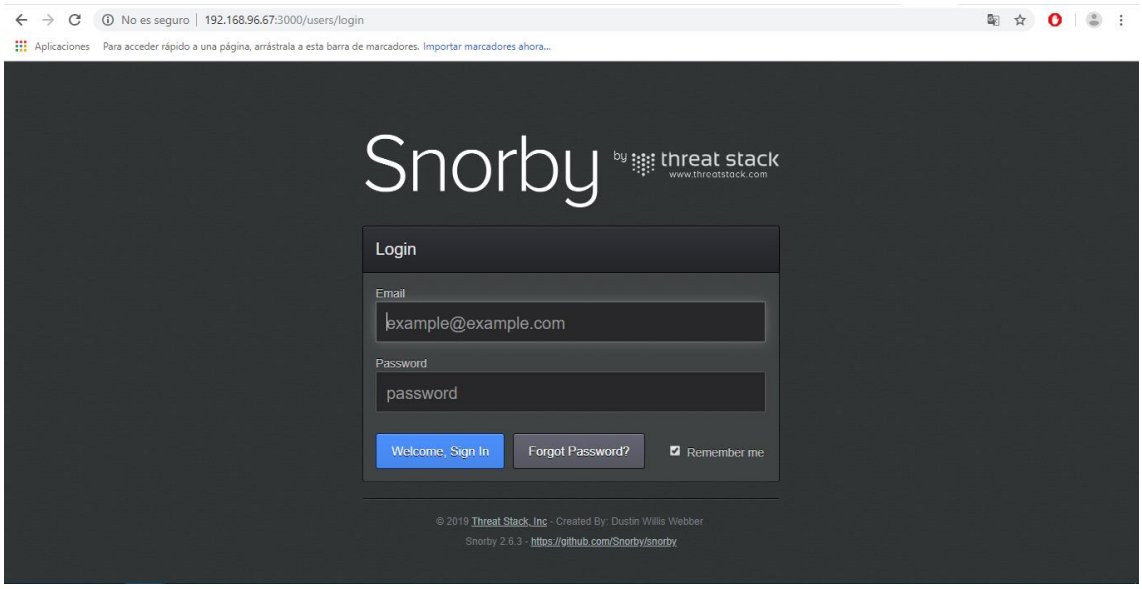


Figura 51. Interfaz gráfica snorby.
Fuente: Elaboración propia (2018).

Ahora que ya tenemos instalado y configurado snorby procederemos a lanzar snorby para que inicie en el puerto 80. Instalaremos Phusion passenger el módulo de de servidor de aplicaciones necesitaremos algunas librerías antes de instalar phusion passenger.

sudo apt-get install -y libcurl4-openssl-dev apache2-threaded-dev libaprutil1-dev libapr1-dev

Antes instalaremos la clave PGP del pasajero y el soporte HTTPS. Con los siguientes comandos

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 561F9B9CAC40B2F7

sudo apt-get install -y apt-transport-https ca-certificates

Agregamos el repositorio de pasajeros con el siguiente comando:

sudo sh -c 'echo deb https://oss-binaries.phusionpassenger.com/apt/passenger wily main > /etc/apt/sources.list.d/passenger.list'

Ahora instalaremos passenger con el siguiente comando.

sudo apt-get install -y libapache2-mod-passenger

Una vez instalado passegner habilitamos el modulo apache y reiniciaremos:

sudo a2enmod passenger

sudo apache2ctl restart

Verificaremos que la instalación haya sido satisfactoria con el siguiente comando sudo /usr/bin/passenger-config validate-install. Figura 52.

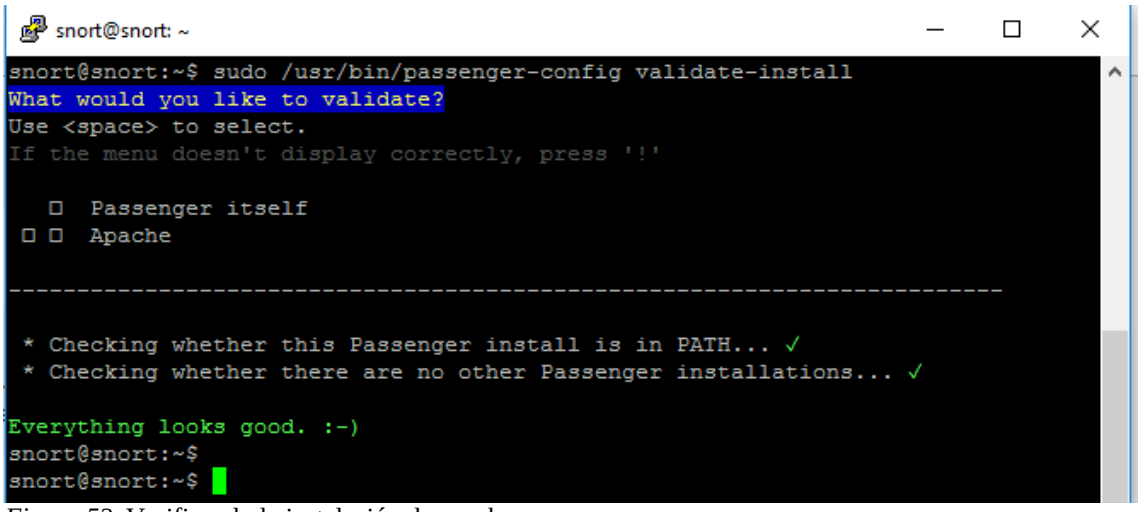


Figura 52. Verificando la instalación de apache y passenger.
Fuente: Elaboración propia (2018).

Verificamos que los modulos de passenger se hayan cargado con el siguiente comando apache2ctl -t -D DUMP_MODULES. Figura 53.

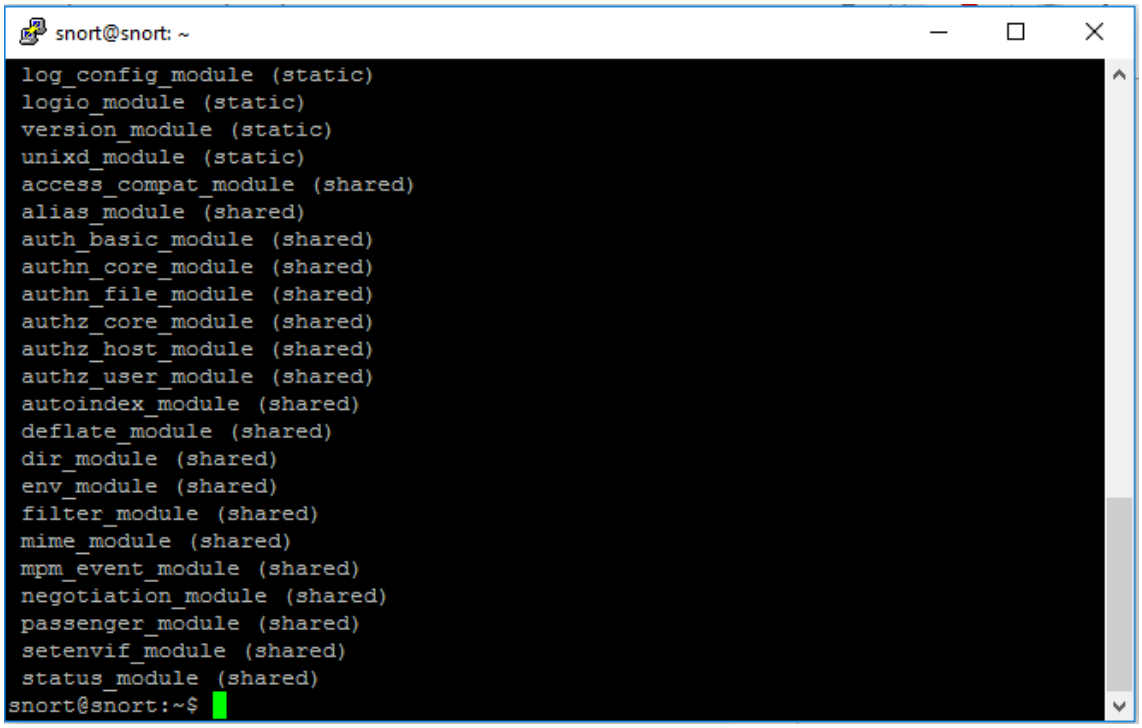


Figura 53. Modulo passenger.
Fuente: Elaboración propia (2018).

Ahora crearemos el sitio web para snorby en la siguiente dirección: sudo nano /etc/apache2/sites-available/snorby.conf ingresamos lo siguiente: Figura 54.

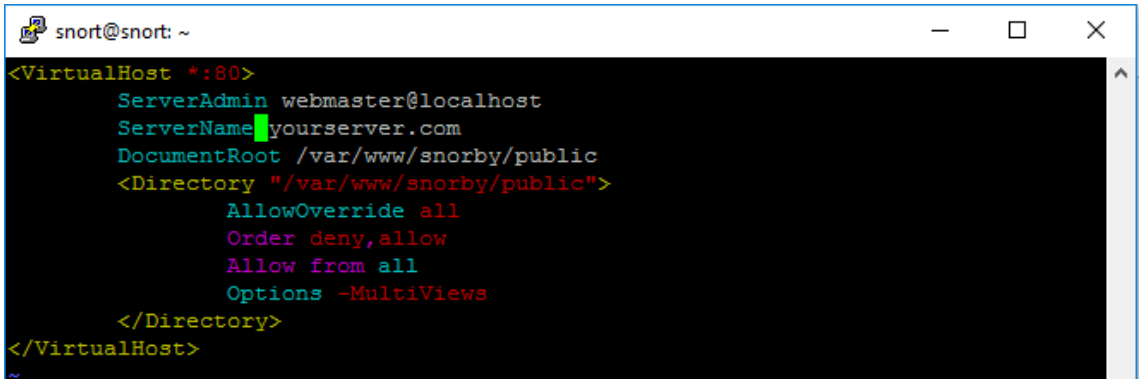


Figura 54. Creando sitio web para snorby.
Fuente: Elaboración propia (2018).

Habilitaremos el sitio y desabilitaremos el sitio predeterminado que estaba y recargaremos apache ingresamos en los directorios respectivos para hacer dicha acción.

cd /etc/apache2/sites-available/

sudo a2ensite snorby.conf

sudo service apache2 reload

cd /etc/apache2/sites-enabled

sudo a2dissite 000-default

sudo service apache2 reload

Ahora editaremos el archivo barnyard2.conf para decirle que genere eventos en la base de datos de snorby que creamos y comentamos la configuración anterior. Figura 55.

sudo nano /etc/snort/barnyard2.conf

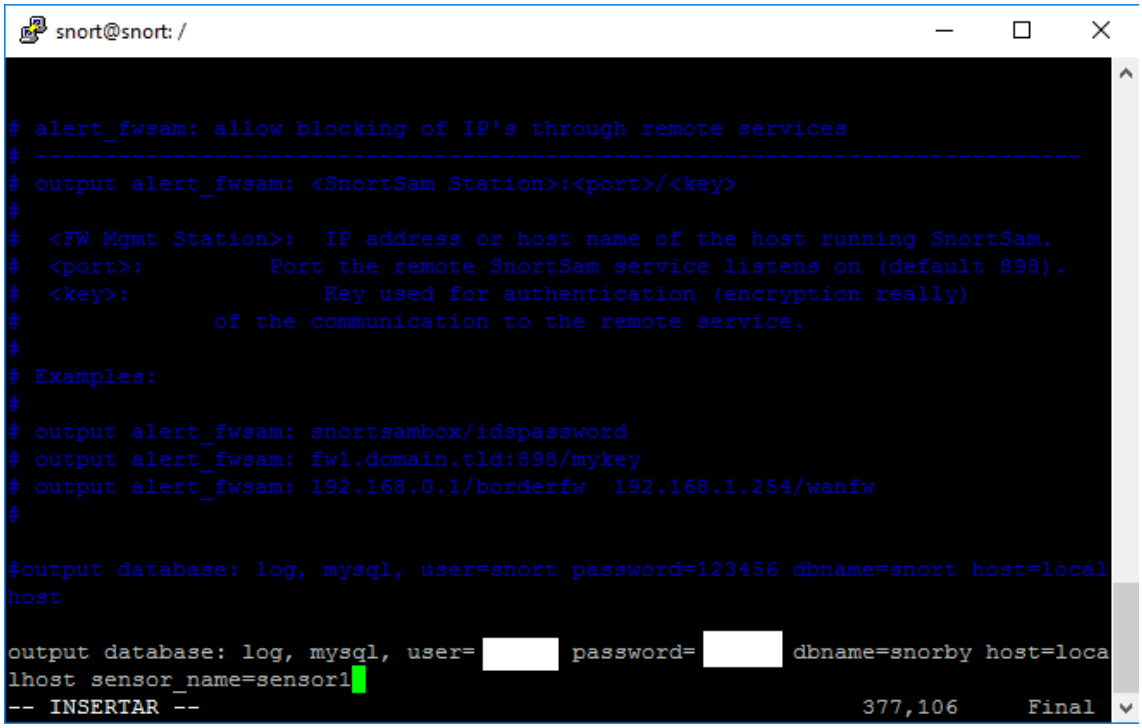


Figura 55. Ingresando la db de snorby en barnyard2.
Fuente: Elaboración propia (2018).

Reiniciamos barnyard2 con el siguiente comando sudo service barnyard2 restart ahora ya tenemos todo configurado, por ultimo crearemos un archivo demonio para snorby. En la siguiente dirección y con el siguiente nombre sudo vim /lib/systemd/system/snorby_worker.service. Ingresamos lo siguiente: Figura 56.

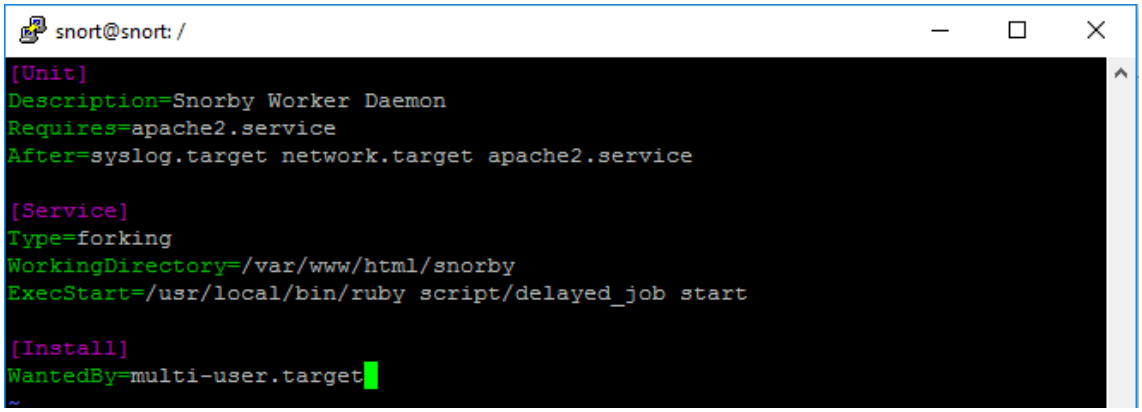


Figura 56. Archivo daemon snorby.
Fuente: Elaboración propia (2018).

Ahora procedemos a ejecutar el script y decirle que se inicie junto con el arranque del sistema.

sudo systemctl enable snorby_worker

systemctl status snorby_worker.service

Una vez ya terminado todo abriremos un navegador web e ingresamos la ruta para iniciar snorby, en la parte del login el usuario y contraseña que nos da por defecto es user: snorby@example.com pass: snorby, una vez ingresado procedemos a cambiar este usuario y contraseña. Figura 57.

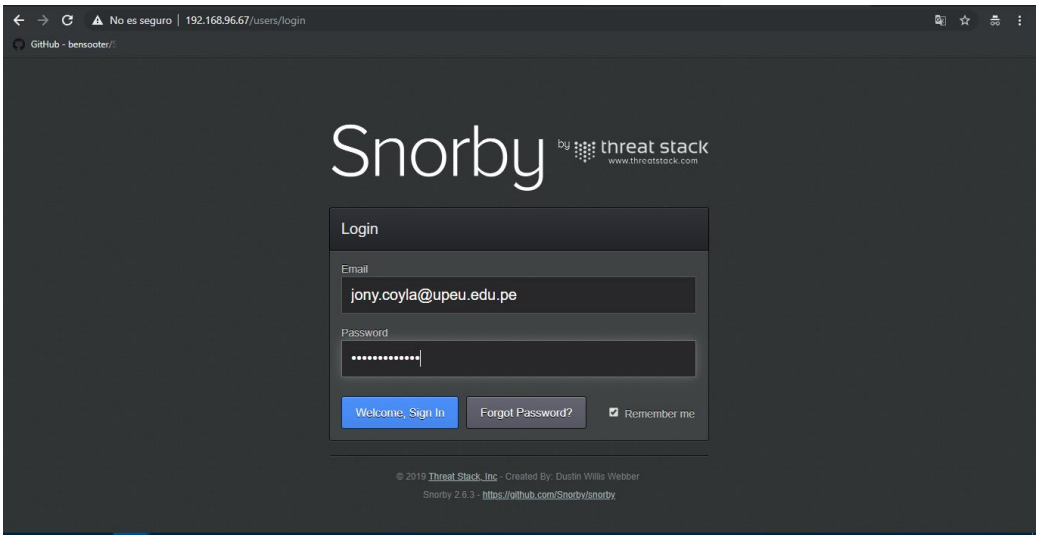


Figura 57. Login snorby.
Fuente: Elaboración propia (2018).

4.11 Monitoreando el tráfico de datos – Medición ISO 27001

Una vez logueado esperamos 10 min para que pueda cargarnos en la pantalla el monitoreo del tráfico, terminado ese lapso de tiempo veremos que nos muestra los eventos que suceden en la red monitoreada. Figura 58.

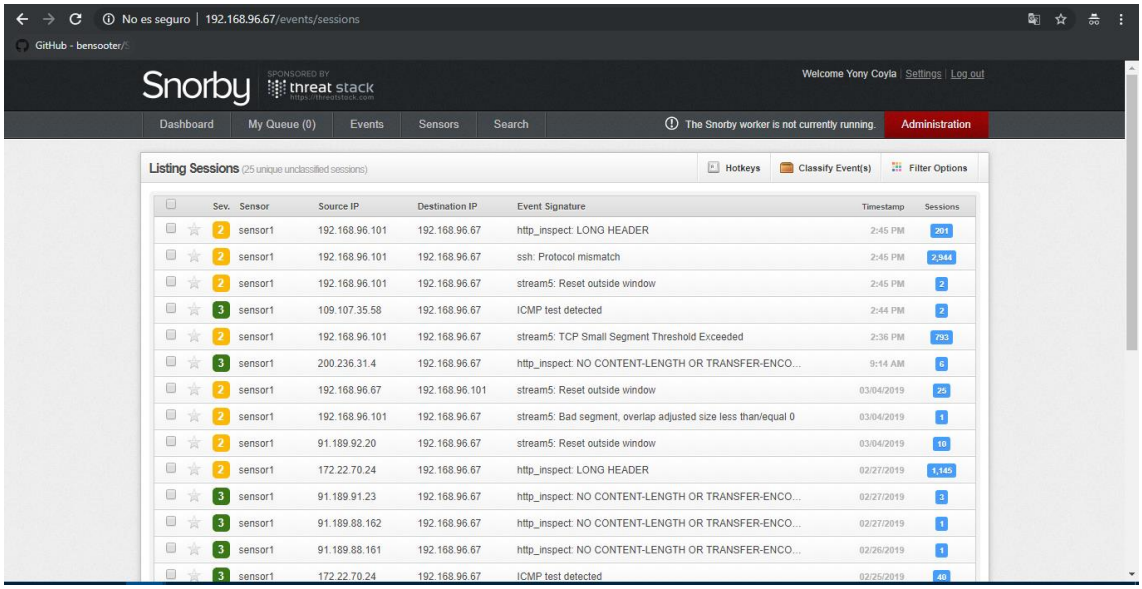


Figura 58. Eventos mostrados por snorby.
Fuente: Elaboración propia (2018).

CAPÍTULO V. Resultados y discusión

5.1 Resultado del objetivo 1

Para la fase de planificación se analisó y diseñó las reglas según el trafico mostrado por los reportes brindados del equipo fortigate el cual cuenta la Universidad Peruana Unón, las reglas mas relevantes que se describieron fueron las de intento de ataque de inyección, sitios web maliciosos, intento de escaneo de puertos y ataques de denegación de servicios DDoS.

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Bad Traffic"; program: snort; content: "Classification|3a| Bad Traffic"; classtype: bad-unknown; normalize; sid: 5000978; rev:5;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Suspected TCP injection"; flow:to_client; window:1; fragbits:!D; sid:1000000; rev:2;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Detection of a Network Scan"; program: snort; content: "Classification: Detection of a Network Scan"; classtype: network-scan; normalize; sid: 5000999; rev:5;)

alert any $EXTERNAL_NET any -> $HOME_NET any (msg:"[SNORT] Denial of Service"; program: snort; content: "Classification|3a| Denial of Service"; classtype: successful-dos; normalize; sid: 5000983; rev:5;)

Podremos ver mas a detalle las reglas descritas, en el capitulo iv de análisis y diseño de reglas. Posteriormente se hizo uso de las reglas que nos brinda snort a travez de un oinkcode ya que nos brinda una gran variedad de reglas de ataques y tráfico malicioso conocidos además de que snort nos brinda gran cantidad de reglas, la comunidad de snort también nos ayuda con reglas que el mismo snort no posee. También tenemos un archivo local en donde guardamos nuestras propias reglas creadas según hayamos visto el tráfico que queramos detectar o bloquear.

5.1.1 Discusión resultado 1

Estos resultados guardan relación con Yauri Lozano (2017) con respecto a la planificación de las reglas propuestas, dicho investigador nos dice que las reglas diseñadas funcionan siempre en cuando esten establecidas y continuamente actualizadas. Dicho resultado es acorde con lo que en esta investigación se halla. Sin embargo en lo que no concuerdo con el autor Lozano es que nos dice que pulled pork y dumpig se encargan de actualizar y detectar errores de dramática, pero según mi punto de vista se debe dar mas énfasis a los requerimientos que la institución brinda al realizar un monitoreo previo.

5.2 Resultado del objetivo 2

En la fase de implementación tenemos la instalación y configuración de nuestro IDS verificaremos que haya sido instalado y configurado correctamente. Para verificar esto se puso una alerta básica, esta alerta consiste en que mi IDS me alerte si alguna maquina hace ping a la red de mi servidor IDS configurado, una vez introducida esta regla se introduce el siguiente comando sudo /usr/local/bin/snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i ens160. Para verificar que en realidad nuestro IDS detecta la regla propuesta. Según la figura 24 podemos ver que nuestra instalación en modo NIDS se hizo correctamente.

Para la instalación y configuración de snort en modo IPS, es básicamente casi la misma instalación que snort en modo NIDS pero con una biblioteca adicional, para esta función necesitaremos la biblioteca especifica de NFQ el cual nos permite trabajar con iptables pero en esta investigación usaremos la configuración con afpacket y monitorear la red con dos interfaces, también realizaremos algunos cambios en la configuración del archivo snort.conf, agregamos en las líneas 169 la configuración de los paquetes de trabajo en modo inline así como nos muestra en la figura 26. De la misma manera insertamos una regla con la opción DROP para bloquear y alertar que alguien está haciendo un ping a la red, snort en modo IPS no solo detecta los paquetes si no que tiene como objetivo bloquear paquetes o ataques, previamente las reglas configuradas.

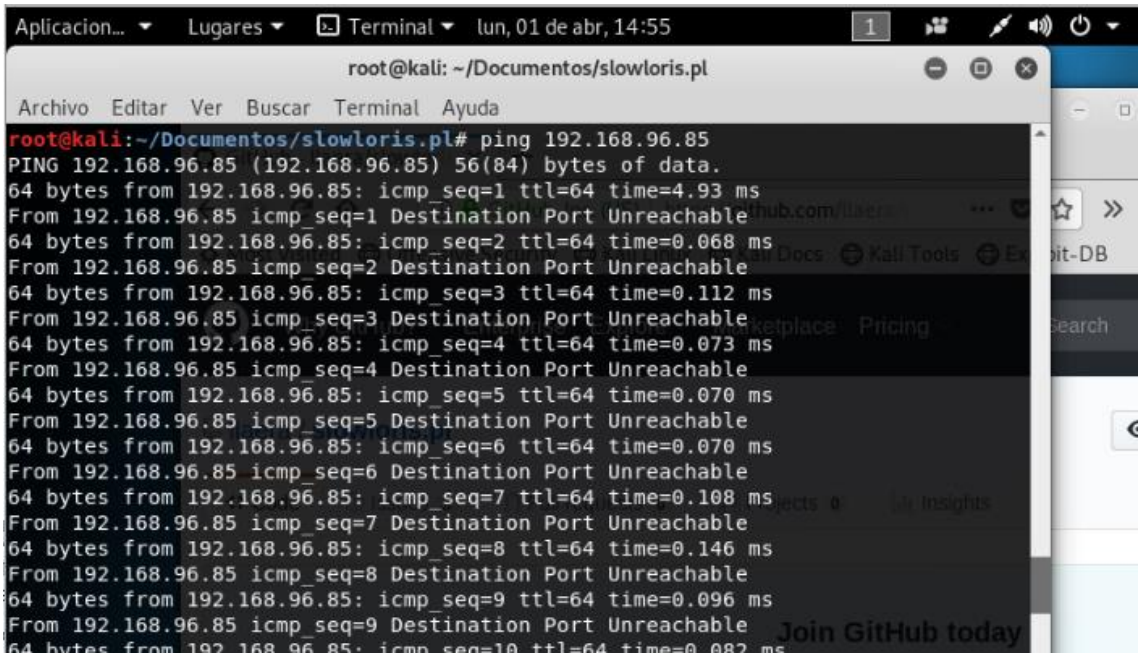


Figura 59. Usuario haciendo ping a ip dentro de la red.
Fuente: Elaboración propia (2018).

En la figura 61 vemos que se realiza un ping a 192.168.96.85 de nuestra red en este caso a la ip de nuestro servidor, también vemos que el tráfico que se espera se empieza a perder, el destino poco a poco es inalcanzable esto ocurre porque en nuestra regla configurada nos dice que bloquee a los usuarios que intenten realizar un ping a nuestra red. En la siguiente Figura 61. Vemos a Snort bloqueando el tráfico con la opción DROP según nuestra regla configurada.

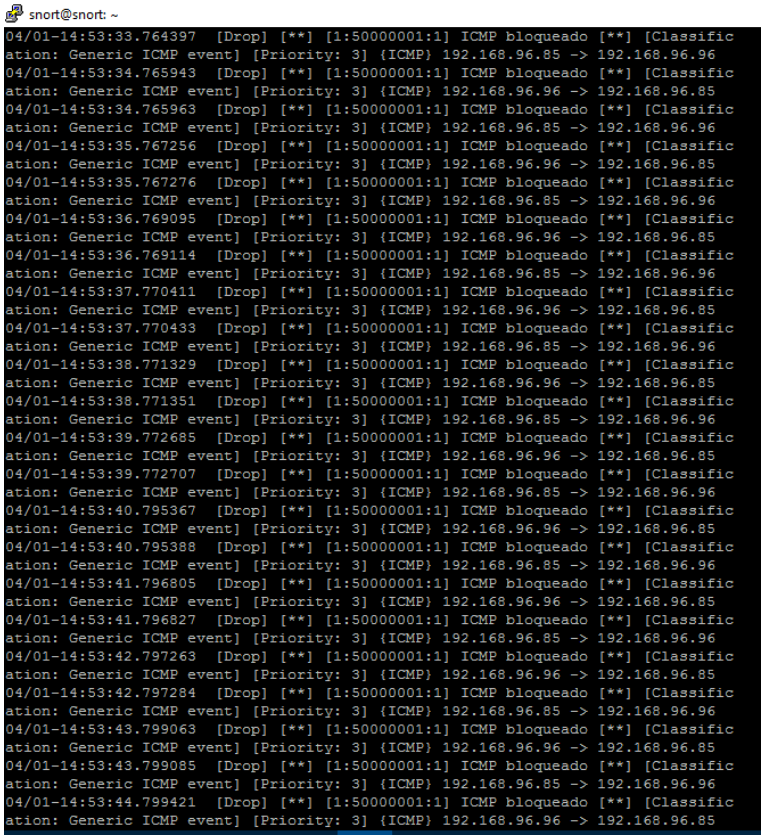


Figura 60. Bloqueo de tráfico en Snort IPS.
Fuente: Elaboración propia (2018).

Una vez realizada las pruebas de funcionamiento de nuestra herramienta, adicionalmente también se agregó el script Pulledpork para la actualización automática de reglas para snort así como también usamos Barnyard2 para el aumento del rendimiento de snort al guardar todo el tráfico en una base de datos msyql y por ultimo añadimos una interfaz gráfica snorby para poder tener una mejor visualización del tráfico de nuestra red.

5.2.1 Discusión resultado 2

Estos resultados guardan relación con Jimenez Alegria (2016) con respecto a la implementación de la instalación y configuración de snort, el investigador nos dice que la forma de tener un mejor monitoreo de un IDS/IPS es con una interfaz web. Dicho resultado es acorde con lo que en esta investigación se halla. Sin embargo en lo que no concuerdo con el autor Jimenez es que menciona que configurar una interfaz web para el IDS/IPS nos ayuda a tener un mejor control y por ende monitoreo del tráfico.

En lo que respecta al monitoreo del IPS podría funcionar simplemente desde la consola ya que la función que tiene es el de bloquear accesos y de esa manera consumir menos recursos de nuestro servidor IDS/IPS.

5.3 Resultado del objetivo 3

Para la fase de medición se evaluó el monitoreo del tráfico en la red de la Universidad Peruana Unión – Filial Juliaca, ahora que ya tenemos todo configurado, nuestras reglas a travez de pulledpork, el uso de barnyard2 para guardar todo en una base de datos y la interfaz gráfica snorby para una mejor visualización pasaremos a interpretar el tráfico en la red de la UPeU.

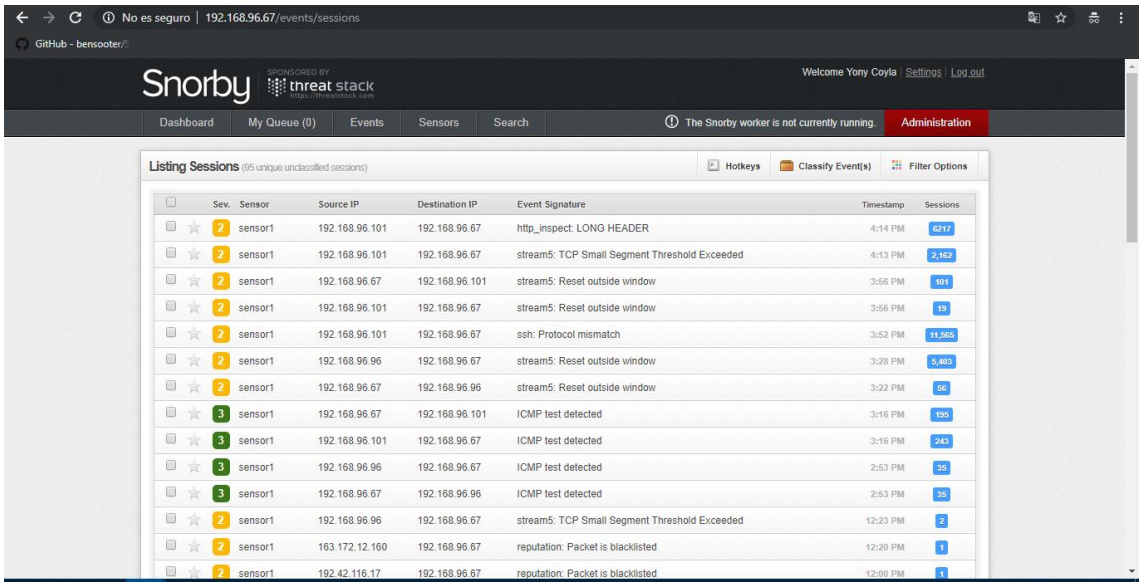


Figura 61. Monitoreo del tráfico en redes internas.
Fuente: Elaboración propia (2018).

En la figura 63 vemos el tráfico de datos interno, snorby nos muestra el resultado del tráfico mediante tres tipos de clasificación, la primera clasificación está compuesta por el tráfico de baja severidad y es de color verde normalmente en esta clasificación es detectada por las reglas básicas compuestas por pocos parámetros en la regla, la siguiente clasificación es la de severidad media y es de color amarillo en esta clasificación snorby detecta el tráfico que intenta burlar la autorización permitida del usuario, algunos ejemplos que tenemos son el escaneo de puertos abiertos, sitios webs con archivos maliciosos o algunos ataques de denegación de servicio (DoS). La ultima clasificación es la de severidad alta de color rojo en esta clasificación nosotros indicamos en nuestras reglas que dicho tráfico sea alertado con una severidad alta pueden ser algunos virus nuevos.

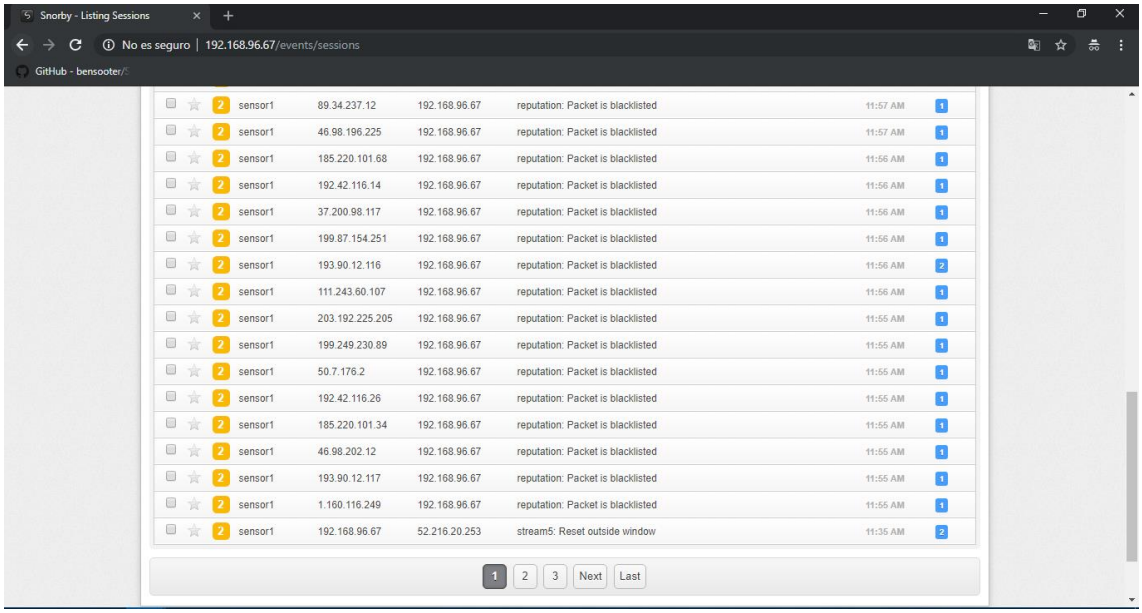


Figura 62. Monitoreo del tráfico de nuestra red hacia redes externas.
Fuente: Elaboración propia (2018).

En la figura 64 vemos el tráfico de la red con redes externas osea los usuarios interactuando con la navegación en internet si nos fijamos en la parte de eventos de firmas vemos que tenemos el siguiente mensaje reputation: packet is blacklisted, esto nos dice que esos paquetes externos están registrados en nuestra lista negra de reglas, existen algunas reglas las cuales se tiene que monitorear detalladamente simplemente por lo desconocido del paquete.

5.3.1 Discusión resultado 3

Estos resultados guardan relación con Carbajal Troya (2015) con respecto a la fase de medición del monitoreo del trafico de la red, el investigador menciona que al monitorizar accesos de alta capacidad hacen que el rendimiento del hardware pueda convertirse en un cuello de botella. Dicho resultado es acorde con lo que en esta investigación se halla. Sin embargo en lo que no concuerdo es, que es cierto que pueda bajar el rendimiento del hardware pero eso pasaría por querer usar la herramienta IDS/IPS para toda la red entonces en lo que respecta esta investigación haciendo uso de la ISO 27001 se propone implementar un IDS/IPS en distintos lugares donde se quiera monitorear el trafico.

5.4 Resultado del objetivo 4

En la fase de mejora tenemos el análisis y el bloqueo de tráfico no permitido en esta fase se entra como en un siclo de constante mejoramiento porque snort no dejara de mejorar

ya sea agregando nueva funcionalidades o configurando algo específico o como también la unión con suricata.

5.4.1 Prueba de ingreso de usuario al servidor

Intentaremos ingresar a nuestro servidor mediante telnet o ssh, se usó una maquina Windows como usuario1 y una máquina virtual kali Linux como usuario2.

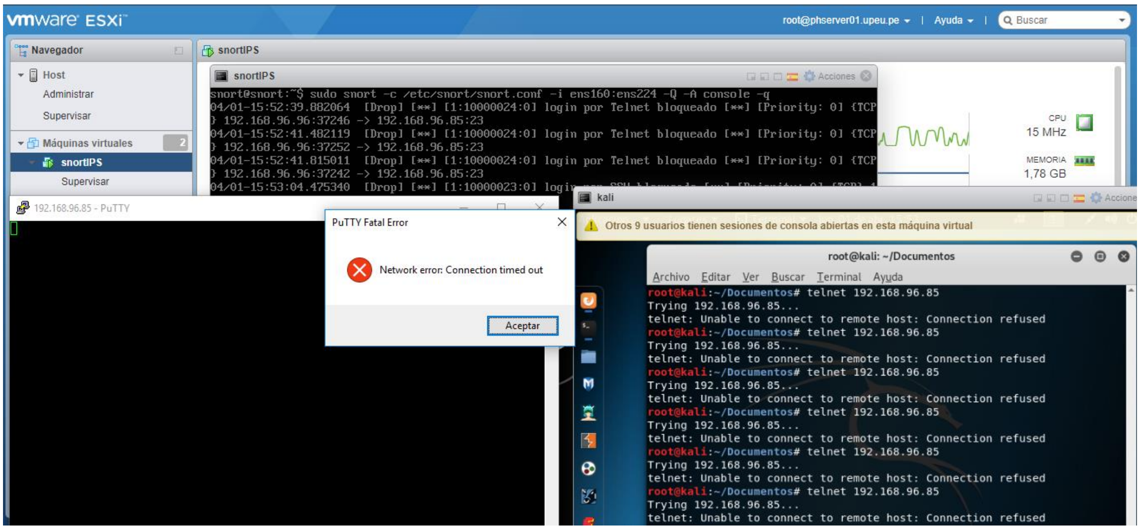


Figura 63. Resultado ingreso por telnet y ssh.
Fuente: Elaboración propia (2018).

En la figura 65. Tenemos como resultado que snort bloquea el ingreso a nuestro servidor mediante ssh y telnet. Para poder ver el mensaje de bloqueo, se abrió la consola de VMware Exsi, en donde está alojado nuestro snort, podemos visualizar que por la herramienta putty del usuario1 la conexión fue rechazada de igual forma para el usuario2 ingresando por telnet nos muestra que la conexión fue rechazada.

5.4.2 Prueba de ataque DDoS

Pondremos a prueba a nuestro snort en modo NIDS desactivando el modo IPS, realizaremos un ataque de denegación de servicio con el programa slowloris el cual nos permite abrumar a un servidor al mantener abiertas muchas conexiones HTTP simultáneamente afectando a apache. Como vemos en la figura 66. Slowloris envía múltiples paquetes al servidor, logrando causar lentitud en el servidor.

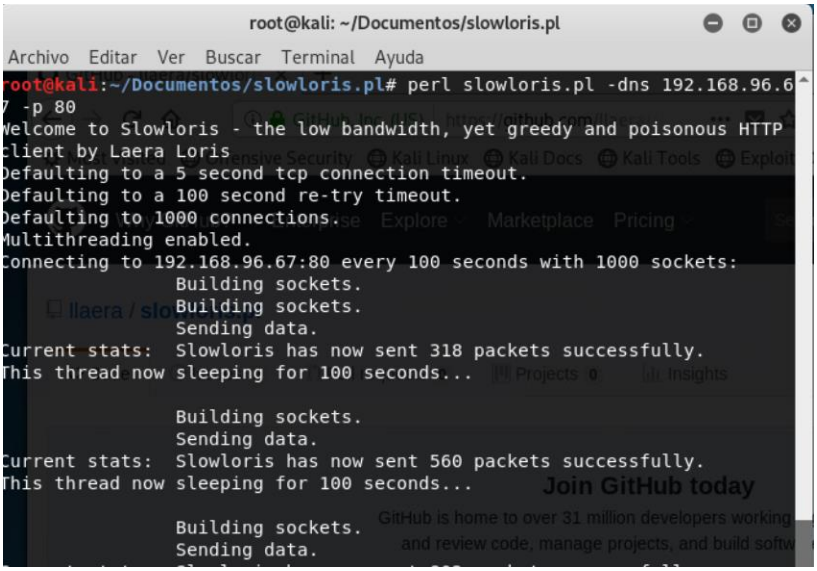


Figura 64. Ataque DDoS con slowloris.
Fuente: Elaboración propia (2018).

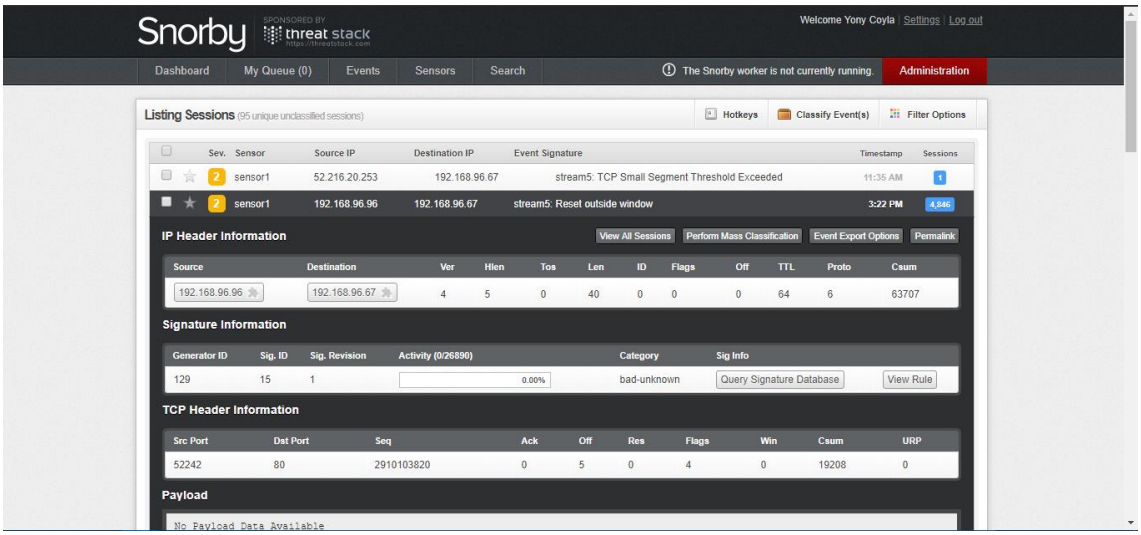


Figura 65. Reporte de Snort en modo NIDS.
Fuente: Elaboración propia (2018).

En la figura 67. Tenemos como resultado que nuestro snort en modo NIDS nos detecta el ataque. Podemos visualizar en el evento el IP origen del ataque como también al IP de destino de ataque también podemos ver que en un solo envio de paquete hubo 4846 peticiones a nuestro servidor.

Ahora pondremos a prueba a nuestro snort en modo IPS iniciándolo, Se realizó el mismo ataque con slowloris. En la figura 68. Tenemos como resultado que el ataque DDoS tuvo un intento fallido ya que si nos fijamos bien slowloris envía 0 paquetes al servidor.

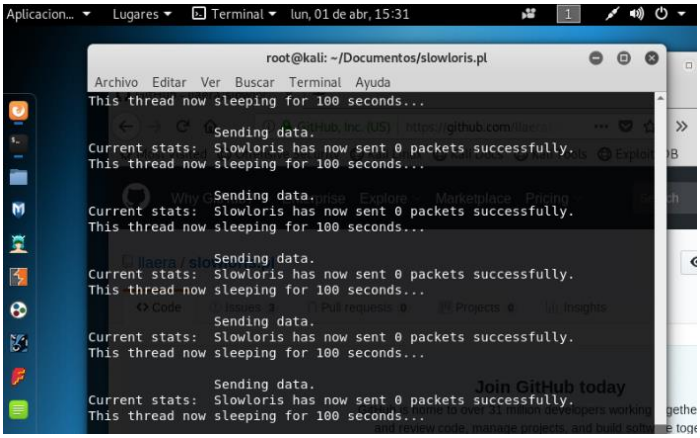


Figura 66. Ataque DDoS con slowloris fallido.
Fuente: Elaboración propia (2018).

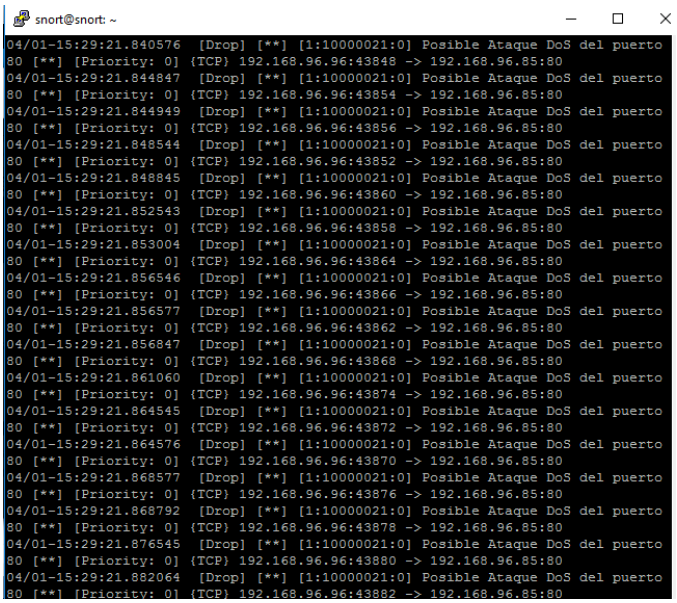


Figura 67. Bloqueo de ataque DDoS por snortIPS.
Fuente: Elaboración propia (2018).

Con la regla configurada para los ataques de DDoS para apache con slowloris, vemos en la figura 69. Que tenemos como resultado que snort IPS bloquea todas las peticiones.

5.4.3 Discusión resultado 4

Estos resultados guardan relación con Jimenez Alegria (2016) con respecto a la fase de mejora haciendo el análisis y las pruebas del bloqueo del trafico no permitido, el investigador menciona que para el caso de ataques de DDoS la configuración que usa es la configuración por defecto de snort. Dicho resultado es acorde con lo que en esta investigación se halla. Sin embargo en lo que no concuerdo es que la configuración por

defecto que nos trae snort esque es muy básica se necesita de ciertos elementos para poder bloquear estos ataques ya que existen diversos tipos de ataques de DDoS.

CAPÍTULO VI. Conclusiones y recomendaciones

6.1 Conclusiones

Con respecto al objetivo general se logró mostrar la utilidad de la herramienta snort con respecto a la seguridad perimetral de la información de la red de la Universidad Peruana Unión. Al monitorear el tráfico y realizar las pruebas necesarias y probar que en realidad nos muestra el resultado requerido, como tambien vemos que el uso de una metodología ISO 27001 mejora la calidad de la implementación de sistemas de seguridad informática por las 4 etapas que tiene las cuales son planificar, implementar, medir y mejorar

En cuanto al primer objetivo específico se logró implementar una gran variedad de reglas mediante el oinkcode que nos brinda snort, estas reglas bloquean ataques como virus, conocidos como también algunos no muy conocidos, gracias a esto solo nos preocupamos de ingresar reglas específicas como de restricción de accesos o de brindar accesos a usuarios específicos.

En cuanto al segundo objetivo específico se logró implementar un sistema de seguridad que pueda permitirnos restringir accesos no deseados así como para mantener la confidencialidad, disponibilidad, integridad y autentificación de la red de la Universidad Peruana Unión.

En cuanto al tercer objetivo específico se logró monitorear con éxito el tráfico de la red de la Universidad Peruana Unión Filial Juliaca. Gracias a esto podremos tener un mejor control de tráfico autorizado y no autorizado. También permitirá al administrador de red estar alerta frente a cualquier tipo de incidente que se pudiera presentar.

Con respecto al cuarto objetivo se logró realizar las pruebas necesarias para verificar la funcionalidad de la herramienta snort como solución de una buena seguridad de la información, realizando pruebas en la red perimetral.

6.2 Recomendaciones

El avance constante de la tecnología hace que estemos necesariamente actualizándonos en esa área, es por eso que se recomienda capacitación constante al personal de la Universidad Peruana Unión Juliaca, sobre los temas de seguridad informática en cuanto a ataques de virus o intrusos o como es que funcionan los paquetes maliciosos, virus y sobre la precaución sobre la información confidencial

Se recomienda la continua mejora del IDS y expandir esta herramienta para configurarlo en distintos lugares de la zona de protección DMZ.

Se recomienda dar un mejor seguimiento a los paquetes desconocidos para así poder crear las reglas respectivas y poder tener actualizadas nuestro conjunto de reglas.

Hacer uso de la norma ISO 27001 con respecto a la seguridad informática es primordial ya no solo nos da etapas para desarrollar algún proyecto si no tambien en la calidad de la implementación del proyecto.

REFERENCIAS

accensit, S. (2016). Seguridad perimetral informática: Información necesaria. Retrieved March 31, 2019, from https://www.accensit.com/blog/seguridad-perimetral-informatica-informacion-necesaria/

Advicera. (2016). ¿Qué es norma ISO 27001? Retrieved October 12, 2018, from https://advisera.com/27001academy/es/que-es-iso-27001/

Belda, N. (2011). Otros usos de Barnyard2 - Security Art Work. Retrieved March 31, 2019, from https://www.securityartwork.es/2011/03/16/otros-usos-de-barnyard2/

Bricata. (2016). Snort, Suricata y Bro: 3 tecnologías de código abierto para proteger redes modernas | Bricata. Retrieved October 22, 2018, from https://bricata.com/blog/snort-suricata-bro-ids/

Carbajal Troya, C. H. (2015). Universidad regional autónoma de los andes " uniandes – ibarra."

Chavarriaga, J., Arboleda, H., & Lidis, G. (2004). Modelo de Investigación en Ingeniería del Software: Una propuesta de investigación tecnológica. *Ingeniería Del Software y Sistemas*. Retrieved from http://www.emn.fr/x-info/harbol07/MIFISIS2004.pdf

Cisco. (2015). Restrictions for Snort IPS, 1–44.

Cisco, N. (2018). Cisco Next-Generation Intrusion Prevention System (NGIPS) - Cisco. Retrieved March 31, 2019, from https://www.cisco.com/c/en/us/products/security/ngips/index.html

Cotoira, F. (2014). Primeros pasos para implementar un IDS con Snort | WeLiveSecurity. Retrieved March 31, 2019, from https://www.welivesecurity.com/la-es/2014/01/13/primeros-pasos-implementacion-ids-snort/

Forcepoint, L. (2018). Integrate McAfee ePO with Forcepoint NGFW. Retrieved from http://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.2.2/GUID-CB6EDFB6-6FB9-4A59-B7FA-CF963046BE61.html

IBM, S. N. I. P. S. (2018). Introducing IBM Security Network Intrusion Prevention System (IPS) products. Retrieved March 31, 2019, from https://www.ibm.com/support/knowledgecenter/en/SSB2MG_4.6.0/com.ibm.ips.doc/concepts/landing_page.htm

ISOTools. (2016). ISO 27001 - Software ISO 27001 de Sistemas de Gestión. Retrieved April 9, 2019, from https://www.isotools.org/normas/riesgos-y-seguridad/iso-27001/

Jimenez Alegria, L. C. (2016). Universidad catolica de santa maria.

LaRepública. (2015). Perú es el quinto país con más ataques cibernéticos en Latinoamérica | LaRepublica.pe. Retrieved April 9, 2019, from https://larepublica.pe/sociedad/714455-peru-es-el-quinto-pais-con-mayor-cantidad-de-ataques-ciberneticos-en-latinoamerica

Llopis, J. (2017). Actualización automática de reglas Snort con PulledPork - Security Art Work. Retrieved March 31, 2019, from https://www.securityartwork.es/2017/01/27/actualizacion-automatica-reglas-snort-pulledpork/

McAfee, L. (2018). Sistema de prevención de intrusiones - Plataforma de seguridad de red | Productos McAfee. Retrieved March 31, 2019, from https://www.mcafee.com/enterprise/en-us/products/network-security-platform.html

McCranie, K. D., Faulkner, M., French, D., Daddis, G. A., Gow, J., & Long, A. (2011). Book Reviews. *Journal of Strategic Studies*, *34*(2), 281–293. https://doi.org/10.1080/01402390.2011.569130

Normas ISO. (2015). ISO 27001 - Seguridad de la información: norma ISO IEC 27001/27002. Retrieved April 1, 2019, from https://www.normas-iso.com/iso-27001/

OpenWebinars. (2015). ¿Qué es el Pentesting? | OpenWebinars.net. Retrieved October 13, 2018, from https://openwebinars.net/blog/que-es-el-pentesting/

Oracle. (2015). MySQL | La base de datos de código abierto más popular del mercado | Oracle España. Retrieved March 31, 2019, from https://www.oracle.com/es/mysql/

Ortego Delgado, D. (2017). Qué es Snort | OpenWebinars. Retrieved March 31, 2019, from https://openwebinars.net/blog/que-es-snort/

PandaSecurity. (2018). ¿A qué se denomina Sistema de Prevención de Intrusos o IPS? - Soporte Técnico Panda Security. Retrieved October 12, 2018, from https://www.pandasecurity.com/usa-es/support/card?id=31452

PortalAndina. (2017). Día de Internet: ciberseguridad en la mira | Especiales | Agencia Peruana de Noticias ANDINA. Retrieved April 9, 2019, from http://portal.andina.pe/edpespeciales/2017/ciberseguridad/index.html

Quadrant. (2019). Enterprise Security, Network Security, Managed Security Services, Managed SIEM, Sagan Technology | Quadrant Information Security. Retrieved April 17, 2019, from https://quadrantsec.com/

Radware, D. (2018). DefensePro DDoS Protection IPS y modelos de protección de

comportamiento. Retrieved March 31, 2019, from https://www.radware.com/products/defensepro-models/

Robalino Diaz, J. W. (2018). Propuestas Metodológica Y Simulación De La Implementación De Un Siem Basado En La Norma Iso 27001 Y/O 27002.

Roesch, M. (2010a). 4. Snort, 1–31.

Roesch, M. (2010b). Introducción, 1–23.

Roesch, M. (2012). Introducción Elementos de Snort.

Ruiz Vieira, K. E., & Delgado Ramos, W. (2018). Universidad de lambayeque facultad de ciencias de ingeniería escuela profesional de ingeniería de sistemas.

Sánchez Márquez, A. (2017). Mejores IDS Opensource para Detección de Intrusiones – Proteger mi PC. Retrieved October 10, 2018, from https://protegermipc.net/2017/02/22/mejores-ids-opensource-deteccion-de-intrusiones/

SeguridadGJSA. (2015). IDS/IPS | seguridadenredesgjsa. Retrieved September 9, 2018, from https://seguridadenredesgjsa.wordpress.com/firewall-y-otros-medios-de-defensa/idsips/

SeguridadYredes. (2019). Snort. Preprocesadores (I) Parte. | Seguridad y Redes. Retrieved March 31, 2019, from https://seguridadyredes.wordpress.com/2009/03/03/snort-preprocesadores-i-parte/

Servidorinfo. (2018). ¿Qué es un ataque DDoS? Retrieved April 7, 2019, from http://www.servidorinfo.info/ataques-ddos/

SGSI. (2016). Norma ISO 27001 y la detección de intrusos en el sistema. Retrieved April 1, 2019, from https://www.pmg-ssi.com/2016/07/como-utilizar-los-sistemas-de-deteccion-de-intrusos-para-cumplir-con-la-norma-iso-27001/

Snort, blog. (2011). Snort Blog: GUIs para Snort. Retrieved March 31, 2019, from https://blog.snort.org/2011/01/guis-for-snort.html

Solarte Martinez, G. R., Castro Bermúdez, Y. V., & Ocampo, C. A. (2017). Sistema de detección de intrusos en redes corporativas. *Scientia et Technica*, *22*(1), 60. https://doi.org/10.22517/23447214.9105

Tiwari, M. (2011). Intrusion Detection System, (April), 39–57. https://doi.org/10.1142/9781848164482_0004

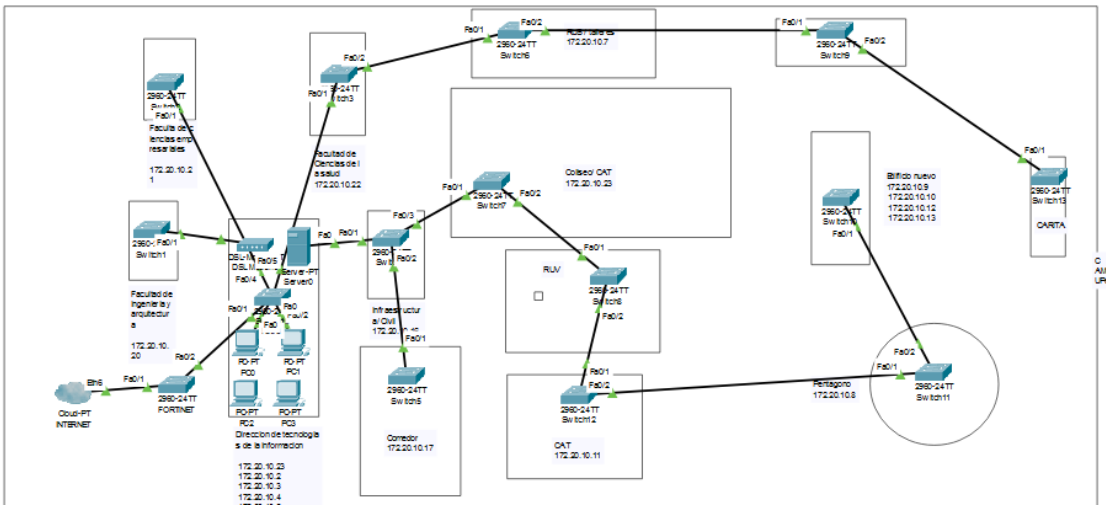
Universidad Internacional de Valencia. (2016). ¿Qué es la seguridad informática y cómo puede ayudarme? | VIU. Retrieved September 9, 2018, from

https://www.universidadviu.es/la-seguridad-informatica-puede-ayudarme/

Yauri Lozano, E. (2017). U NIVERSIDAD N ACIONAL DE T UCUMAN F acultad de C iencias N aturales e I nstituto M iguel L illo T esista ı M aría J osefina R uiz.

ANEXOS

Anexo A. Arquitectura antes de snort IDS/IPS



Anexo B. Reporte de Fortinet sobre el análisis de amenazas a la UPeU- FJ de la fecha 2018/08/11

2018-08-11 00:00 - 2018-08-12 00:00

Summary Report

Threat Analysis

Top Threats

Threat	Category	Level	Score	%
Failed Connection Attempt	Firewall Control	Low	63955	23.6%
api.apiv6.com	Malicious Websites	High	39900	14.7%
event.apiv5.com	Malicious Websites	High	28380	10.5%
mega.co.nz	urlfilter	High	25380	9.4%
cdn.livestream.com	Internet Radio and TV	High	21750	8.0%
cpos.wtfast.com	Games	High	16260	6.0%
wg.spotify.com	Internet Radio and TV	High	14070	5.2%
bittorrent	p2p	Low	11245	4.1%
mobilecrush.king.com	Games	High	10440	3.9%
cpos2.wtfast.com	Games	High	10230	3.8%
psiphon	proxy	Medium	8320	3.1%
analytics.rayjump.com	Games	High	8100	3.0%
bittorrent	p2p	Low	4665	1.7%
evantative.top	Malicious Websites	High	4200	1.5%
proxy.http	proxy	Medium	1660	0.6%
freegate.searching	proxy	Medium	570	0.2%
okhttp.library.vpn	proxy	Medium	550	0.2%
D-Link.DSL-2750B.CLI.OS.Command.Injection	Attack	Critical	250	0.1%
Adware/AirPush!Android	Malware	Critical	200	0.1%
STUNSHELL.Web.Shell.Remote.Code.Execution	Attack	Medium	160	0.1%
MS.IIS.WebDAV.PROPFIND.ScStoragePathFromUrl.Buf...rflow	Attack	Critical	150	0.1%
D-Link.DSL-2750B.CLI.OS.Command.Injection	Attack	Critical	150	0.1%

Anexo C. Reporte de Fortinet sobre el análisis de amenazas a la UPeU- FJ de la fecha 2018/08/13

2018-08-12 00:00 - 2018-08-13 00:00

Summary Report

Threat Analysis

Top Threats

Threat	Category	Level	Score	%
	Malicious Websites	High	67740	23.9%
Failed Connection Attempt	Firewall Control	Low	34520	12.2%
psiphon	proxy	Medium	31800	11.2%
event.apiv5.com	Malicious Websites	High	30180	10.6%
api.apiv6.com	Malicious Websites	High	27960	9.9%
play.google.com	Freeware and Software Downloads	High	26610	9.4%
wg.spotify.com	Internet Radio and TV	High	17220	6.1%
replay251.valve.net	Games	High	14940	5.3%
mega.co.nz	urlfilter	High	13980	4.9%
www.dota2.com	Games	High	5850	2.1%
bittorrent	p2p	Low	4680	1.6%
analytics.rayjump.com	Games	High	2910	1.0%
cloud.netflix.com	urlfilter	High	1740	0.6%
proxy.http	proxy	Medium	1070	0.4%
bittorrent	p2p	Low	925	0.3%
Failed Connection Attempt	Firewall Control	Low	555	0.2%
okhttp.library.vpn	proxy	Medium	340	0.1%
D-Link.DSL-2750B.CLI.OS.Command.Injection	Attack	Critical	250	0.1%
D-Link.DSL-2750B.CLI.OS.Command.Injection	Attack	Critical	200	0.1%
W32/Farfli.CEN!tr	Malware	Critical	50	0.0%
Bash.Function.Definitions.Remote.Code.Execution	Attack	Critical	50	0.0%
tor	proxy	Medium	40	0.0%

Anexo D. Reporte de Fortinet sobre el análisis de amenazas a la UPeU- FJ de la fecha 2019/01/17

2019-01-16 00:00 - 2019-01-17 00:00

Summary Report

Threat Analysis

Top Threats

Threat	Category	Level	Score	%
Failed Connection Attempt	Firewall Control	Low	215255	47.0%
ayce.gameloft.com	Games	High	70140	15.3%
wg.spotify.com	Internet Radio and TV	High	58620	12.8%
metaservices.microsoft.com	Games	High	53850	11.8%
sniper3d.fungames-forfree.com	Games	High	18930	4.1%
play.google.com	Freeware and Software Downloads	High	8220	1.8%
proxy.googlezip.net	Proxy Avoidance	High	5120	1.1%
cloud.rovio.com	Games	High	4410	1.0%
mega.co.nz	urlfilter	High	4050	0.9%
farmheroesmobile.king.com	Games	High	3330	0.7%
settings-ssl.xboxlive.com	Games	High	2520	0.5%
proxy.http	proxy	Medium	2370	0.5%
rc.appcloudbox.net	Phishing	High	2280	0.5%
l.adsbk.com	Phishing	High	1740	0.4%
bob-janus.gameloft.com	Games	High	1740	0.4%
bob-iris.gameloft.com	Games	High	1350	0.3%
event.apiv8.com	Malicious Websites	High	1290	0.3%
mini5.opera-mini.net	Proxy Avoidance	High	1040	0.2%
apresolve.spotify.com	Internet Radio and TV	High	990	0.2%
rtep.msgamestudios.com	Games	High	990	0.2%
			Total: 458235	

Anexo E. Reporte de Fortinet sobre el análisis de amenazas a la UPeU- FJ de la fecha 2019/02/01

2019-01-31 00:00 - 2019-02-01 00:00

Summary Report

Threat Analysis

Top Threats

Threat	Category	Level	Score	%
Failed Connection Attempt	Firewall Control	Low	286195	57.1%
ayce.gameloft.com	Games	High	116010	23.1%
wg.spotify.com	Internet Radio and TV	High	46230	9.2%
proxy.googlezip.net	Proxy Avoidance	High	14720	2.9%
mega.co.nz	urlfilter	High	6270	1.3%
mobilecrush.king.com	Games	High	3540	0.7%
proxy.http	proxy	Medium	3520	0.7%
usa.mobhey.com	Games	High	3240	0.6%
candycrushsodamobile.king.com	Games	High	2790	0.6%
bittorrent	p2p	Low	2660	0.5%
bob-ets-wsg.gameloft.com	Games	High	2430	0.5%
sg01.rayjump.com	Games	High	2070	0.4%
bob-janus.gameloft.com	Games	High	1920	0.4%
event.apiv8.com	Malicious Websites	High	1650	0.3%
cdn.exoticads.com	Other Adult Materials	High	1440	0.3%
prod-video-sa-east-1.pscp.tv	Internet Radio and TV	High	1410	0.3%
www.wildtangent.com	Games	High	1380	0.3%
replay251.valve.net	Games	High	1350	0.3%
data.ero-advertising.com	Pornography	High	1330	0.3%
adserver.juicyads.com	Pornography	High	1330	0.3%
			Total: 501485	

Anexo F. Reporte de Fortinet sobre el análisis de amenazas a la UPeU- FJ de la fecha 2019/02/14

2019-02-13 00:00 - 2019-02-14 00:00

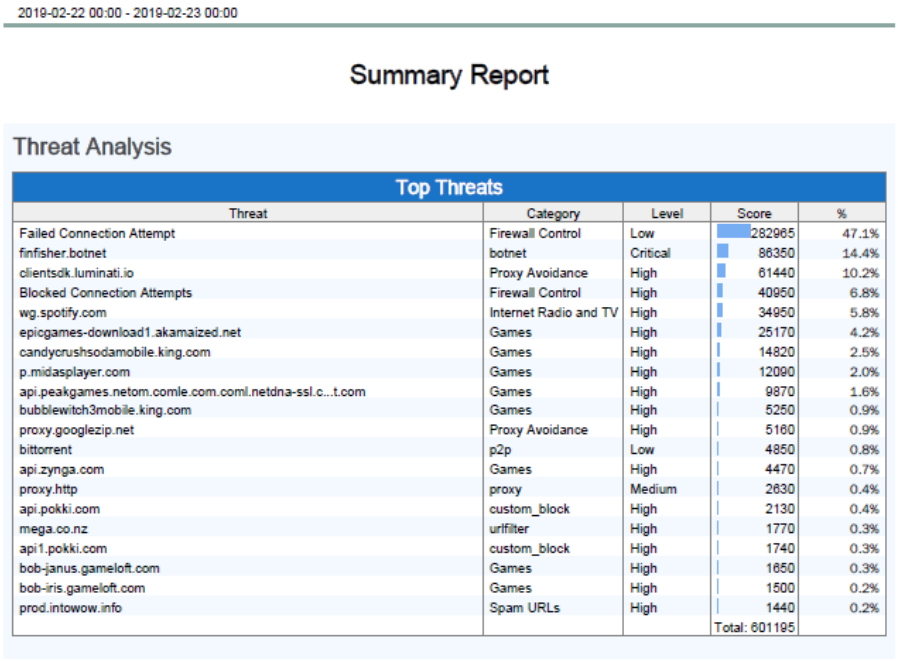
Summary Report

Threat Analysis

Top Threats

Threat	Category	Level	Score	%
Failed Connection Attempt	Firewall Control	Low	206990	43.2%
finfisher.botnet	botnet	Critical	81550	17.0%
wg.spotify.com	Internet Radio and TV	High	43740	9.1%
play.google.com	Freeware and Software Downloads	High	31170	6.5%
rtep.msgamestudios.com	Games	High	18630	3.9%
ayce.gameloft.com	Games	High	12420	2.6%
92.63.197.48	Malicious Websites	High	8100	1.7%
bittorrent	p2p	Low	7775	1.6%
ougohoueahgoushughoej.in	Malicious Websites	High	7320	1.5%
112.126.94.107	Malicious Websites	High	6600	1.4%
123.56.228.49	Malicious Websites	High	6300	1.3%
eoufaoeuhoauengi.biz	Malicious Websites	High	5820	1.2%
bellsyscdn.com	Malicious Websites	High	5700	1.2%
95.153.31.22	Malicious Websites	High	5640	1.2%
92.63.197.60	Malicious Websites	High	5340	1.1%
iuefgauiaiduihgs.in	Malicious Websites	High	5280	1.1%
iuefgauiaiduihgs.net	Malicious Websites	High	5220	1.1%
proxy.googlezip.net	Proxy Avoidance	High	5160	1.1%
booomaahuuoooapl.com	Malicious Websites	High	5160	1.1%
maeobnaoefhgoajo.com	Malicious Websites	High	5160	1.1%
			Total: 479075	

Anexo G. Reporte de Fortinet sobre el análisis de amenazas a la UPeU- FJ de la fecha 2019/02/23



2019-02-22 00:00 - 2019-02-23 00:00

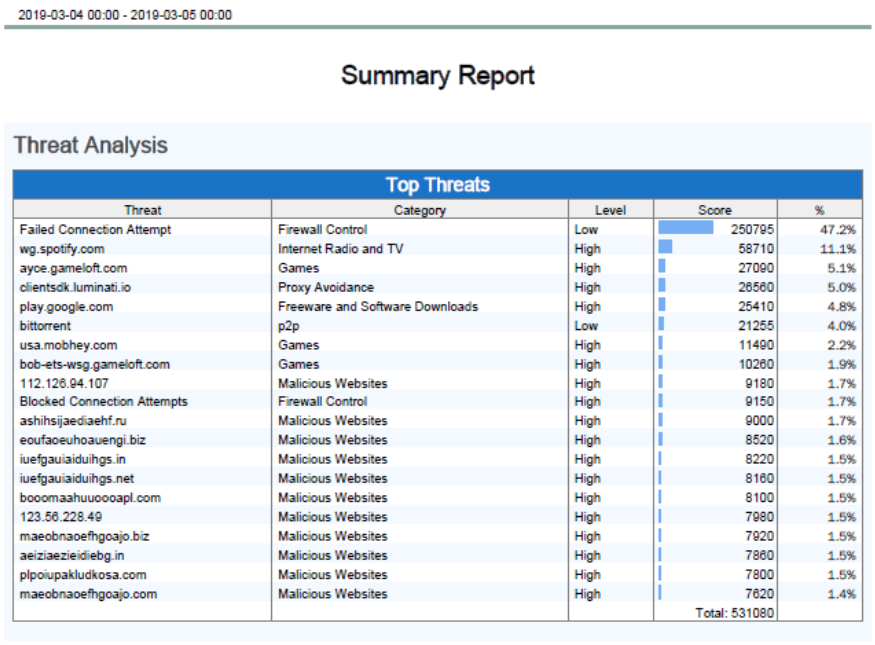
Summary Report

Threat Analysis

Top Threats

Threat	Category	Level	Score	%
Failed Connection Attempt	Firewall Control	Low	282965	47.1%
finfisher.botnet	botnet	Critical	86350	14.4%
clientsdk.luminati.io	Proxy Avoidance	High	61440	10.2%
Blocked Connection Attempts	Firewall Control	High	40950	6.8%
wg.spotify.com	Internet Radio and TV	High	34950	5.8%
epicgames-download1.akamaized.net	Games	High	25170	4.2%
candycrushsodamobile.king.com	Games	High	14820	2.5%
p.midasplayer.com	Games	High	12090	2.0%
api.peakgames.netom.comle.com.coml.netdna-ssl.c...t.com	Games	High	9870	1.6%
bubblewitch3mobile.king.com	Games	High	5250	0.9%
proxy.googlezip.net	Proxy Avoidance	High	5160	0.9%
bittorrent	p2p	Low	4850	0.8%
api.zynga.com	Games	High	4470	0.7%
proxy.http	proxy	Medium	2630	0.4%
api.pokki.com	custom_block	High	2130	0.4%
mega.co.nz	urlfilter	High	1770	0.3%
api1.pokki.com	custom_block	High	1740	0.3%
bob-janus.gameloft.com	Games	High	1650	0.3%
bob-iris.gameloft.com	Games	High	1500	0.2%
prod.intowow.info	Spam URLs	High	1440	0.2%
			Total: 601195	

Anexo H. Reporte de Fortinet sobre el análisis de amenazas a la UPeU- FJ de la fecha 2019/03/05



2019-03-04 00:00 - 2019-03-05 00:00

Summary Report

Threat Analysis

Top Threats

Threat	Category	Level	Score	%
Failed Connection Attempt	Firewall Control	Low	250795	47.2%
wg.spotify.com	Internet Radio and TV	High	58710	11.1%
ayce.gameloft.com	Games	High	27090	5.1%
clientsdk.luminati.io	Proxy Avoidance	High	26560	5.0%
play.google.com	Freeware and Software Downloads	High	25410	4.8%
bittorrent	p2p	Low	21255	4.0%
usa.mobhey.com	Games	High	11490	2.2%
bob-ets-wsg.gameloft.com	Games	High	10260	1.9%
112.126.94.107	Malicious Websites	High	9180	1.7%
Blocked Connection Attempts	Firewall Control	High	9150	1.7%
ashihsijaediaehf.ru	Malicious Websites	High	9000	1.7%
eoufaoeuhoauengi.biz	Malicious Websites	High	8520	1.6%
iuefgauiaiduihgs.in	Malicious Websites	High	8220	1.5%
iuefgauiaiduihgs.net	Malicious Websites	High	8160	1.5%
booomaahuuoooapl.com	Malicious Websites	High	8100	1.5%
123.56.228.49	Malicious Websites	High	7980	1.5%
maeobnaoefhgoajo.biz	Malicious Websites	High	7920	1.5%
aeiziaeziediebg.in	Malicious Websites	High	7860	1.5%
plpoiupakludkosa.com	Malicious Websites	High	7800	1.5%
maeobnaoefhgoajo.com	Malicious Websites	High	7620	1.4%
			Total: 531080	

Anexo I. Constancia de autorizacion del area de DTI para realizar el proyecto de tesis

dti DIRECCIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN
Somos parte de TI

El que suscribe, Sub Director de la Dirección de Tecnologías de la Información de la Universidad Peruana Unión Sede Juliaca, Dr. Jorge Alejandro Sánchez Garcés:

AUTORIZA la realización del proyecto de tesis: ***"Implementación de un modelo de un sistema de detección y prevención de intrusos (IDS/IPS), basado en la norma ISO 27001, para el monitoreo perimetral de la seguridad informática, en la red de la Universidad Peruana Unión - Filial Juliaca"*** en nuestras instalaciones del área de Redes y Conectividad, a realizarse por el Bachiller Yony Coyla Jarita.

Se expide la presente autorización a solicitud del interesado.

Juliaca 07 de enero de 2019

DIRECCION GENERAL DE SISTEMAS · UNIVERSIDAD PERUANA UNION · JULIACA · PERU

Dr. Jorge Alejandro Sánchez Garcés
Sub Director de la Dirección de Tecnologías de la Información