

UNIVERSIDAD DE ALMERÍA

COMPLEMENTO TRABAJO FIN DE GRADO

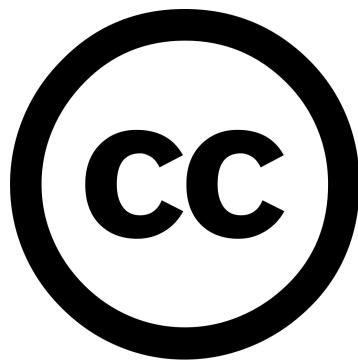
# **Utilización de un NIDS para redes SOHO (R-Snort)**

Autor: Petrovics, Deian Orlando

Tutor: Gómez López, Julio

Cotutor: Padilla Soriano, Nicolás

24 de marzo de 2025



Este trabajo está bajo una licencia Creative Commons  
Atribución-NoComercial-CompartirIgual 4.0 Internacional.

# Índice general

<b>Abreviaturas</b>	4
<b>Introducción</b>	5
<b>1 Motivación</b>	6
<b>2 Objetivos</b>	7
<b>3 Fases de la realización y cronograma</b>	8
<b>4 Estructura y metodología</b>	9
<b>5 Sistemas de Detección de Intrusos</b>	10
5.1 Los sistemas de detección de intrusos	10
5.1.1 Definición y propósito de los IDS	10
5.1.2 Diferencia entre detección de intrusos y prevención de intrusos	10
5.1.3 Tipos de IDS	10
5.1.4 Basados en análisis de comportamiento vs. basados en firma	11
5.1.5 Arquitectura y componentes de un IDS	11
5.1.6 Funcionalidades clave	11
5.1.7 Ventajas y limitaciones de los IDS	11
5.1.8 Contexto actual y dificultades	12
5.1.9 Ejemplos y casos prácticos	12
5.2 Snort	12
5.2.1 Introducción a Snort	12
5.2.2 Características principales	12
5.2.3 Arquitectura de Snort	13
5.2.4 Reglas y actualizaciones	13
5.2.5 Personalización y extensibilidad	13
5.2.6 Casos de uso y ejemplos	14
5.2.7 Limitaciones y consideraciones	14
5.3 Necesidades de seguridad en pequeñas redes	14
5.3.1 Definición de pequeñas redes (SOHO)	14
5.3.2 Amenazas y vulnerabilidades comunes	15
5.3.3 Desafíos específicos	15
5.3.4 Necesidades clave de seguridad	15
5.3.5 Soluciones adecuadas para SOHO	16
5.3.6 Beneficios de implementar un NIDS en pequeñas redes	16

5.3.7	Buenas prácticas de seguridad para SOHO . . . . .	16
5.4	Complementos y plugins para Snort . . . . .	16
<b>6</b>	<b>Utilización de un NIDS para pequeñas redes . . . . .</b>	<b>18</b>
6.1	Introducción . . . . .	18
6.2	Especificaciones, características y requisitos . . . . .	18
6.2.1	Especificaciones de Snort 3 . . . . .	18
6.2.2	Requisitos del sistema . . . . .	19
6.2.3	Hardware (Raspberry Pi 5) . . . . .	19
6.2.4	Software (Snort y sus complementos) . . . . .	19
6.3	Instalación y personalización de complementos . . . . .	21
6.3.1	Instalación de Snort V3 . . . . .	21
6.3.2	Instalación de reglas y plugins . . . . .	36
6.3.3	Configuración preprocesador HTTP Inspect . . . . .	38
6.3.4	SSL Inspector . . . . .	40
6.3.5	Stream IP . . . . .	40
6.3.6	Stream TCP . . . . .	41
6.3.7	Reputation . . . . .	41
6.3.8	Datos sensibles . . . . .	42
6.3.9	Antivirus ClamAV . . . . .	43
6.4	Generación de un script para la instalación automática . . . . .	44
<b>7</b>	<b>Casos prácticos: utilización de R-Snort . . . . .</b>	<b>45</b>
7.1	Entorno de trabajo . . . . .	45
7.2	Instalación . . . . .	45
7.3	Utilización y pruebas . . . . .	45
7.3.1	Benchmark de rendimiento . . . . .	45
7.3.2	Pruebas . . . . .	45
7.4	Resumen . . . . .	45
<b>Resultados y discusión . . . . .</b>	<b>46</b>	
<b>Conclusiones . . . . .</b>	<b>47</b>	
<b>Bibliografía . . . . .</b>	<b>48</b>	
<b>A Anexo A . . . . .</b>	<b>49</b>	
<b>B Anexo B . . . . .</b>	<b>50</b>	

# Abreviaturas

(Todavía en proceso)

- **NIDS:** Network Intrusion Detection System
- **SOHO:** Small Office/Home Office
- **R-Snort:** Versión de Snort para Raspberry Pi

# Introducción

La creciente dependencia de la tecnología en el día a día ha resaltado la importancia de garantizar la seguridad en las redes informáticas, independientemente de su tamaño. Sin embargo, las pequeñas redes, como las utilizadas en oficinas y hogares, suelen estar desprotegidas debido a la falta de recursos técnicos y económicos, lo que las convierte en objetivos vulnerables para posibles ataques. Este trabajo propone una solución accesible y práctica que permita a estos entornos mejorar significativamente su nivel de seguridad mediante el desarrollo de un sistema de detección de intrusos (IDS) basado en herramientas de bajo coste y fácil implementación.

# **1. Motivación**

Desde el inicio de mi formación, sentí una especial inclinación hacia la seguridad informática y la administración de sistemas. Asignaturas como Administración de Redes, Sistemas Operativos y Seguridad Informática me permitieron profundizar en los retos a los que se enfrentan redes pequeñas y medianas, evidenciando que, a diferencia de las grandes empresas, estos entornos carecen de medidas de protección adecuadas. Esta realidad despertó en mí el interés de buscar soluciones prácticas y asequibles para mitigar esta problemática, con el objetivo de hacer la seguridad informática más accesible para estos entornos.

## 2. Objetivos

El principal objetivo de este trabajo es diseñar y desarrollar un sistema de detección de intrusos (IDS) adaptado a pequeñas redes, empleando Snort y una Raspberry Pi. Los objetivos específicos serán:

- **Integración de herramientas complementarias:** Incorporar plugins, antivirus y filtros de contenido para cubrir las necesidades de seguridad específicas de pequeñas redes.
- **Automatización y portabilidad:** Crear un script automatizado que facilite la instalación y configuración del sistema en distintos entornos.
- **Evaluación de efectividad y rendimiento:** Realizar pruebas para garantizar que el sistema sea eficiente y adecuado para su propósito en condiciones reales.

Se trata proporcionar una solución sencilla, económica y efectiva que impulse la seguridad en redes de menor escala.

### **3. Fases de la realización y cronograma**

(EN DESARROLLO)

## **4. Estructura y metodología**

(EN DESARROLLO)

# **5. Sistemas de Detección de Intrusos**

## **5.1. Los sistemas de detección de intrusos**

El objetivo de esta sección es mostrar una visión general de los Sistemas de Detección de Intrusos (IDS), su importancia en la seguridad informática y cómo encajan en el panorama actual de ciberseguridad.

### **5.1.1. Definición y propósito de los IDS**

Un Sistema de Detección de Intrusos (IDS) es una herramienta de seguridad que monitorea redes y sistemas en busca de actividades maliciosas o violaciones de políticas de seguridad. Su función principal es identificar accesos no autorizados o comportamientos anómalos que puedan comprometer la integridad, confidencialidad o disponibilidad de los sistemas informáticos.

### **5.1.2. Diferencia entre detección de intrusos y prevención de intrusos**

Mientras que un IDS se encarga de detectar y alertar sobre posibles intrusiones, un Sistema de Prevención de Intrusos (IPS) no solo detecta, sino que también toma medidas para bloquear o prevenir dichas intrusiones en tiempo real.

### **5.1.3. Tipos de IDS**

#### **Basados en host (HIDS)**

Monitorizan y analizan la actividad interna de un sistema individual, revisando logs, integridad de archivos y procesos en ejecución.

#### **Basados en red (NIDS)**

Supervisan el tráfico de red en busca de actividades sospechosas, analizando paquetes que circulan por la red para detectar patrones de ataque.

### 5.1.4. Basados en análisis de comportamiento vs. basados en firma

#### Anomaly-based

Detectan desviaciones del comportamiento normal establecido, identificando actividades inusuales que podrían indicar una intrusión.

#### Signature-based

Buscan patrones conocidos de ataques comparando el tráfico con una base de datos de firmas de amenazas previamente identificadas.

### 5.1.5. Arquitectura y componentes de un IDS

Un IDS típico consta de:

- **Sensores:** Capturan datos de la red o del host.
- **Analizadores:** Procesan y examinan los datos recopilados para identificar posibles intrusiones.
- **Bases de datos de firmas:** Almacenan patrones de ataques conocidos para la detección basada en firmas.
- **Interfaz de gestión:** Permite a los administradores configurar el sistema, revisar alertas y generar informes.

### 5.1.6. Funcionalidades clave

- **Monitoreo en tiempo real:** Supervisa continuamente la actividad para detectar intrusiones de manera inmediata.
- **Generación de alertas:** Notifica a los administradores sobre actividades sospechosas o confirmadas.
- **Registro de eventos:** Documenta incidentes para análisis posterior y cumplimiento normativo.
- **Integración con otros sistemas de seguridad:** Trabaja conjuntamente con firewalls, sistemas de gestión de eventos (SIEM) y otras herramientas para una defensa más robusta.

### 5.1.7. Ventajas y limitaciones de los IDS

#### Ventajas

- Detección temprana de amenazas.
- Mejora en la respuesta a incidentes.
- Cumplimiento de normativas de seguridad.

### Limitaciones

- Posibilidad de falsos positivos y negativos.
- Necesidad de actualización constante para reconocer nuevas amenazas.
- Consumo de recursos y posible impacto en el rendimiento del sistema.

#### 5.1.8. Contexto actual y dificultades

Con el incremento de amenazas avanzadas y persistentes (APT), los IDS deben evolucionar incorporando inteligencia artificial y aprendizaje automático para mejorar la detección. Además, la integración con sistemas de gestión de información y eventos de seguridad (SIEM) es esencial para una respuesta coordinada y eficaz.

#### 5.1.9. Ejemplos y casos prácticos

Ataques como el gusano *SQL Slammer* en 2003 fueron detectados por IDS basados en firmas, mientras que amenazas más sofisticadas, como *Stuxnet* en 2010, requirieron análisis más avanzados para su identificación. Comparaciones entre HIDS y NIDS en entornos específicos muestran que, aunque los HIDS ofrecen una visión detallada del host, los NIDS proporcionan una perspectiva más amplia del tráfico de red.

## 5.2. Snort

En esta sección se pretende profundizar en Snort como uno de los NIDS más populares y versátiles, destacando sus características, funcionamiento y relevancia en entornos de seguridad.

### 5.2.1. Introducción a Snort

Desarrollado por Martin Roesch en 1998, Snort ha evolucionado hasta convertirse en un estándar de facto en sistemas de detección de intrusiones de código abierto. En 2013, Cisco Systems adquirió Sourcefire, la empresa detrás de Snort, integrándolo en su portafolio de soluciones de seguridad.

### 5.2.2. Características principales

- **Motor de detección basado en firmas:** Utiliza reglas para identificar patrones de ataque conocidos.
- **Modos de operación:**
  - **Sniffer:** Captura y muestra paquetes en tiempo real.
  - **Packet Logger:** Guarda paquetes en disco para análisis posterior.
  - **Network Intrusion Detection:** Analiza tráfico y detecta intrusiones.

### 5.2.3. Arquitectura de Snort

- **Preprocesadores:** Preparan el tráfico para su análisis, manejando tareas como la normalización y la desfragmentación.
- **Motor de detección:** Aplica reglas al tráfico para identificar posibles amenazas.
- **Plugins de salida:** Definen cómo se reportan las alertas, ya sea a través de archivos, bases de datos o syslog.

### 5.2.4. Reglas y actualizaciones

#### Estructura de una regla de Snort

Una regla típica de Snort se compone de dos partes principales, el encabezado y las opciones. Por un lado el encabezado define la acción a tomar (por ejemplo, alertar), el protocolo, las direcciones IP de origen y destino, y los puertos. Las opciones especifican condiciones adicionales, como patrones de contenido a buscar, mensajes de alerta y otros parámetros.

#### Tipos de reglas

- **Community Rules:** Reglas creadas y mantenidas por la comunidad de usuarios de Snort, disponibles de forma gratuita bajo la licencia GPLv2.
- **Registered Rules:** Reglas proporcionadas por Cisco Talos, disponibles para usuarios registrados con un retraso de 30 días respecto a su lanzamiento para suscriptores.
- **Subscriber Rules:** Reglas actualizadas en tiempo real, disponibles para suscriptores de pago, ofreciendo protección contra amenazas emergentes sin retrasos.

#### Gestión de reglas

Herramientas como *PulledPork* y *Oinkmaster* facilitan la descarga, actualización y gestión de las reglas de Snort, automatizando procesos y asegurando que el sistema esté protegido contra las últimas amenazas.

### 5.2.5. Personalización y extensibilidad

#### Creación de reglas personalizadas

Los administradores pueden desarrollar reglas específicas adaptadas a las necesidades particulares de su entorno, permitiendo una detección más precisa de amenazas relevantes para su organización.

#### Uso de variables y listas

Snort permite definir variables para representar direcciones IP, rangos de puertos y otras configuraciones, facilitando la gestión y actualización de las reglas.

### Integración con otros sistemas

Snort puede integrarse con herramientas como *Barnyard2*, *Snorby* y *Sguil* para mejorar la gestión de alertas, análisis de eventos y generación de informes.

### 5.2.6. Casos de uso y ejemplos

#### Implementación en diferentes entornos

Snort es utilizado en una amplia variedad de escenarios, desde pequeñas oficinas hasta grandes corporaciones y entornos educativos, gracias a su capacidad de adaptación.

#### Detección de ataques comunes

Snort puede identificar actividades maliciosas como inyecciones SQL, escaneos de puertos y ataques de denegación de servicio (DoS), proporcionando alertas en tiempo real para una respuesta rápida.

### 5.2.7. Limitaciones y consideraciones

#### Rendimiento en redes de alta velocidad

En entornos con tráfico de red intenso, es importante contar con hardware adecuado y una configuración óptima para asegurar que Snort funcione correctamente sin afectar el rendimiento.

#### Gestión de falsos positivos

La función de ajustar y afinar las reglas para minimizar las alertas falsas es prioritario, evitando así una sobrecarga de información y permitiendo a los administradores centrarse en amenazas reales.

## 5.3. Necesidades de seguridad en pequeñas redes

Se pretende analizar las necesidades específicas de seguridad en redes pequeñas, como oficinas domésticas o pequeñas empresas (SOHO), y cómo soluciones como Snort pueden ser implementadas eficientemente.

### 5.3.1. Definición de pequeñas redes (SOHO)

Las redes SOHO suelen estar compuestas por un número reducido de dispositivos y recursos limitados, y a menudo carecen de personal especializado en TI. A pesar de su tamaño, estas redes juegan un papel importante para la economía y requieren medidas de seguridad adecuadas.

### 5.3.2. Amenazas y vulnerabilidades comunes

#### Amenazas externas

Incluyen malware, phishing y ataques de fuerza bruta dirigidos a servicios expuestos a Internet.

#### Amenazas internas

Pueden surgir del uso inapropiado de recursos, dispositivos no seguros conectados a la red y falta de políticas de seguridad claras.

#### Dispositivos IoT

La proliferación de dispositivos IoT introduce nuevas vulnerabilidades debido a configuraciones inseguras y falta de actualizaciones.

### 5.3.3. Desafíos específicos

#### Limitaciones de recursos

Presupuestos reducidos y ausencia de personal especializado dificultan la implementación de soluciones de seguridad robustas.

#### Conectividad constante

La dependencia de servicios en la nube y el teletrabajo aumentan la superficie de ataque y requieren medidas de seguridad adicionales.

### 5.3.4. Necesidades clave de seguridad

#### Protección básica

Implementación de firewalls, antivirus y mantenimiento de software actualizado.

#### Monitoreo y detección

Es esencial conocer la actividad en la red; herramientas como Snort pueden proporcionar visibilidad y detección de amenazas.

#### Facilidad de uso y gestión

Las soluciones deben ser intuitivas y, preferiblemente, automatizar tareas como actualizaciones y generación de informes.

### 5.3.5. Soluciones adecuadas para SOHO

#### Herramientas de código abierto

Ofrecen ventajas en costo y flexibilidad, además de contar con comunidades de soporte activas.

#### Dispositivos de bajo costo

Hardware asequible, como *Raspberry Pi*, puede utilizarse para implementar soluciones de seguridad efectivas.

#### Servicios gestionados

Externalizar ciertas funciones de seguridad puede ser una opción viable para pequeñas empresas sin personal especializado.

### 5.3.6. Beneficios de implementar un NIDS en pequeñas redes

- **Detección temprana de amenazas:** Permite identificar y responder rápidamente a actividades maliciosas.
- **Visibilidad del tráfico de red:** Proporciona información sobre el comportamiento de la red y posibles vulnerabilidades.
- **Cumplimiento de normativas básicas de seguridad:** Ayuda a cumplir con estándares y regulaciones aplicables.

### 5.3.7. Buenas prácticas de seguridad para SOHO

- **Políticas de contraseñas fuertes:** Implementar contraseñas robustas y cambiarlas periódicamente.
- **Segmentación de red:** Separar segmentos de red para limitar el alcance de posibles intrusiones.
- **Educación y concienciación de usuarios:** Capacitar al personal sobre prácticas seguras y reconocimiento de amenazas comunes.
- **Copias de seguridad regulares:** Realizar backups periódicos para asegurar la recuperación ante incidentes.

## 5.4. Complementos y plugins para Snort

Listado de los más importantes:

- Actualización
- Antivirus

- Snort Inline
- Filtrado de contenido

# 6. Utilización de un NIDS para pequeñas redes

## 6.1. Introducción

Un Sistema de Detección de Intrusos en la Red (NIDS, por sus siglas en inglés) es una herramienta de seguridad que permite la monitorización del tráfico en busca de actividades sospechosas o ataques. En el caso de pequeñas redes, un NIDS puede proporcionar una primera línea de defensa sin la necesidad de costosas soluciones empresariales.

En este documento, se describe la instalación y configuración de Snort 3 y complementos como NIDS en una Raspberry Pi 5 con Ubuntu Server, aprovechando su eficiencia en el consumo de recursos y su capacidad de detección de amenazas en redes pequeñas.

## 6.2. Especificaciones, características y requisitos

### 6.2.1. Especificaciones de Snort 3

Snort 3 es una versión mejorada de su predecesor, con una arquitectura modular y mejoras significativas en rendimiento y configuración. Algunas de sus características clave incluyen:

- **Arquitectura modular:** permite una configuración flexible y personalizable.
- **Mejoras en rendimiento:** optimizaciones en el motor de detección para un procesamiento más eficiente.
- **Soporte para multihilo:** permite el uso de múltiples procesadores para mejorar el rendimiento en sistemas modernos.
- **Reglas en Lua:** facilita una configuración más avanzada y personalizable.
- **Integración con DAQ (Data Acquisition Library):** ofrece opciones flexibles para la captura de paquetes en la red.
- **Compatibilidad con reglas de Snort 2:** permite utilizar reglas preexistentes.

### 6.2.2. Requisitos del sistema

Para implementar Snort 3 en una Raspberry Pi 5 con Ubuntu Server, se necesita:  
(Estos no son los requisitos, es una mezcla de requisitos con las especificaciones de mi Raspberry Pi 5, en futuras versiones se mejorará)

- **Sistema Operativo:** Ubuntu Server (64-bit) compatible con ARM.
- **Procesador:** ARM Cortex-A76 de la Raspberry Pi 5.
- **Memoria RAM:** Mínimo 4 GB recomendados para un rendimiento óptimo.
- **Almacenamiento:** SSD externo o microSD de al menos 16 GB.
- **Interfaz de red:** Ethernet (se recomienda deshabilitar WiFi para mayor seguridad).

El entorno de trabajo define los componentes físicos y lógicos involucrados en la implementación del NIDS con Snort 3. Para estructurar este apartado, se consideran los siguientes elementos:

### 6.2.3. Hardware (Raspberry Pi 5)

La Raspberry Pi 5 se utiliza como base para la implementación del NIDS debido a su bajo consumo energético y su potencia suficiente para analizar tráfico de red en pequeños entornos. Sus características clave incluyen:

- **CPU:** ARM Cortex-A76 (4 núcleos a 2.4 GHz)
- **GPU:** VideoCore VII
- **RAM:** Modelos de 8GB LPDDR4X
- **Conectividad:**
  - 2 puertos USB 3.0
  - 2 puertos USB 2.0
  - 1 puerto Ethernet Gigabit
  - WiFi 802.11ac y Bluetooth 5.0 (deshabilitados por seguridad)
- **Almacenamiento:**
  - MicroSD 128GB
  - Soporte para SSD a través de USB 3.0

### 6.2.4. Software (Snort y sus complementos)

Snort 3 es un sistema de prevención y detección de intrusos en la red (NIDS/-NIPS) que introduce una serie de mejoras significativas sobre sus versiones anteriores, incluyendo mayor eficiencia, modularidad y una arquitectura basada en plugins. Estas mejoras hacen que Snort 3 sea más adaptable, eficiente y personalizable.

## Comparación con Snort 2

Snort 3 introduce una serie de mejoras sobre Snort 2:

- Configuración más flexible y simplificada.
- Mayor rendimiento gracias al uso de múltiples hilos.
- Soporte para más de 200 plugins que permiten ampliar su funcionalidad.
- Sistema de reglas más eficiente y simplificado.
- Mejora en la detección de amenazas emergentes y reducción de falsos positivos.

## Plugins y complementos utilizados

Para potenciar las capacidades de detección de Snort 3 en este proyecto, se han seleccionado los siguientes preprocesadores y herramientas adicionales:

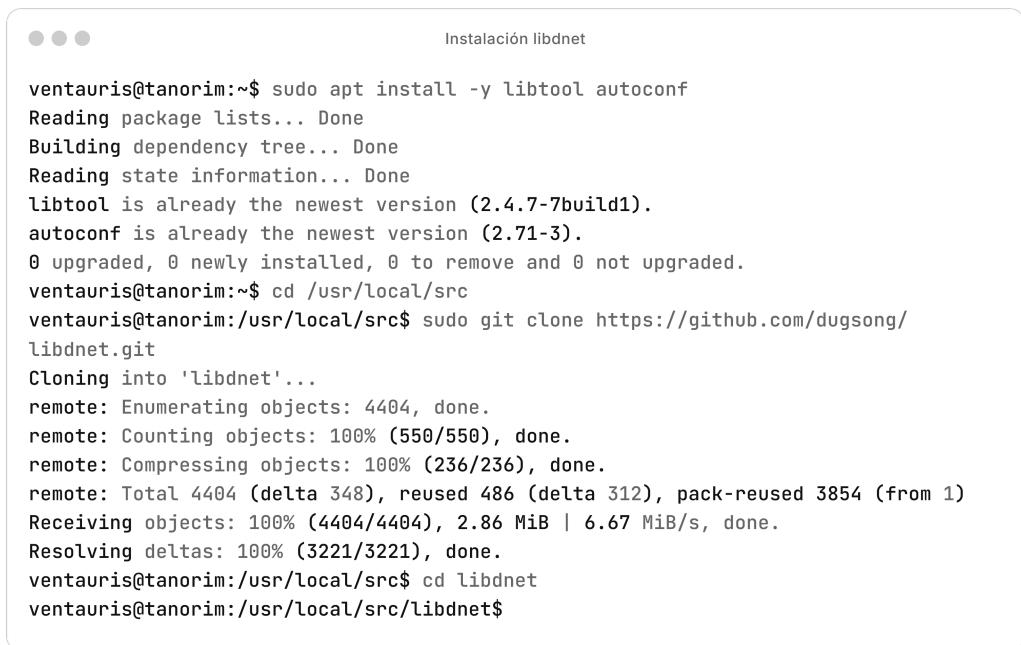
- **HTTP Inspect:** Analiza tráfico HTTP/HTTPS para detectar ataques SQL/XSS e irregularidades en los encabezados.
- **SSL/TLS:** Inspecciona metadatos del tráfico cifrado y detecta conexiones sospechosas.
- **Frag3:** Ensambla paquetes IP fragmentados para detectar intentos de evasión. (Obsoleto, sustituto encontrado y configurado correctamente)
- **Stream5:** Reensambla flujos TCP/UDP para un análisis más preciso. (Obsoleto, sustituto encontrado y configurado correctamente)
- **Reputation Preprocessor:** Bloquea tráfico de fuentes maliciosas basado en listas de reputación.
- **Sensitive Data Preprocessor:** Detecta información sensible como números de tarjetas de crédito o credenciales expuestas. (Obsoleto, sustituto encontrado y configurado correctamente)
- **Barnyard2:** Procesa registros de Snort y almacena alertas en bases de datos sin afectar el rendimiento. (Obsoleto no agregado, se buscará sustituto)
- **Pulled Pork:** Gestiona la actualización automática de reglas desde fuentes oficiales como Talos y Emerging Threats. (Obsoleto no agregado, se buscará sustituto)
- **ClamAV:** Sistema antivirus de código abierto que complementa la detección de amenazas de Snort con análisis basado en firmas.

## 6.3. Instalación y personalización de complementos

### 6.3.1. Instalación de Snort V3

El desarrollo de lo que será R-Snort empezará con la instalación de Snort mismo en su versión actual. Para ello, es buena práctica actualizar repositorios y el equipo mediante un ‘update’ y un ‘upgrade’ antes de comenzar la instalación. Una vez hecho esto, empezaremos con la instalación de las distintas librerías y dependencias de Snort V3.

Primero, instalamos un par de herramientas necesarias (libtool y autoconf) para compilar Snort 3. Luego, nos movemos al directorio /usr/local/src y clonamos el repositorio de libdnet desde GitHub, que es una librería importante para que Snort funcione correctamente. Finalmente, entramos en el directorio libdnet para seguir con la instalación.



```
Instalación libdnet

ventauris@tanorim:~$ sudo apt install -y libtool autoconf
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
libtool is already the newest version (2.4.7-7build1).
autoconf is already the newest version (2.71-3).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ventauris@tanorim:~$ cd /usr/local/src
ventauris@tanorim:/usr/local/src$ sudo git clone https://github.com/dugsong/libdnet.git
Cloning into 'libdnet'...
remote: Enumerating objects: 4404, done.
remote: Counting objects: 100% (550/550), done.
remote: Compressing objects: 100% (236/236), done.
remote: Total 4404 (delta 348), reused 486 (delta 312), pack-reused 3854 (from 1)
Receiving objects: 100% (4404/4404), 2.86 MiB | 6.67 MiB/s, done.
Resolving deltas: 100% (3221/3221), done.
ventauris@tanorim:/usr/local/src$ cd libdnet
ventauris@tanorim:/usr/local/src/libdnet$
```

Figura 6.1: Instalando dependencias necesarias.

Ahora instalamos el paquete check, que es una herramienta para ejecutar pruebas en C. Es un requisito para compilar libdnet correctamente. Con esto, seguimos preparando el entorno antes de compilar Snort 3.



```
Instalación check

ventauris@tanorim:/usr/local/src/libdnet$ sudo apt install -y check
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
.
.
.
ventauris@tanorim:/usr/local/src/libdnet$
```

Figura 6.2: Instalandoo dependencia check.

Aquí ejecutamos `./configure` para preparar la compilación de libdnet. Este script revisa el entorno del sistema, verifica dependencias y configura los archivos necesarios para compilar el código correctamente.



```
Instalación check

ventauris@tanorim:/usr/local/src/libdnet$ sudo ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a race-free mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
.
.
.
ventauris@tanorim:/usr/local/src/libdnet$
```

Figura 6.3: Configurando libdnet antes de la compilación.

Posteriormente ejecutamos sudo make para compilar libdnet. Este comando convierte el código fuente en ejecutables y bibliotecas listas para su instalación. Vemos que recorre los directorios del proyecto, asegurándose de que todos los archivos necesarios sean procesados.



Instalación check

```
ventauris@tanorim:/usr/local/src/libdnet$ sudo make
Making all in include
make[1]: Entering directory '/usr/local/src/libdnet/include'
make  all-recurse
make[2]: Entering directory '/usr/local/src/libdnet/include'
Making all in dnet
make[3]: Entering directory '/usr/local/src/libdnet/include/dnet'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/usr/local/src/libdnet/include/dnet'
make[3]: Entering directory '/usr/local/src/libdnet/include'
make[3]: Leaving directory '/usr/local/src/libdnet/include'
make[2]: Leaving directory '/usr/local/src/libdnet/include'
make[1]: Leaving directory '/usr/local/src/libdnet/include'
Making all in man
.
.
.
make[2]: Leaving directory '/usr/local/src/libdnet/test/dnet'
make[2]: Entering directory '/usr/local/src/libdnet/test'
make[2]: Nothing to be done for 'all-am'.
make[2]: Leaving directory '/usr/local/src/libdnet/test'
make[1]: Leaving directory '/usr/local/src/libdnet/test'
make[1]: Entering directory '/usr/local/src/libdnet'
make[1]: Nothing to be done for 'all-am'.
make[1]: Leaving directory '/usr/local/src/libdnet'
ventauris@tanorim:/usr/local/src/libdnet$
```

Figura 6.4: Compilando libdnet con make.

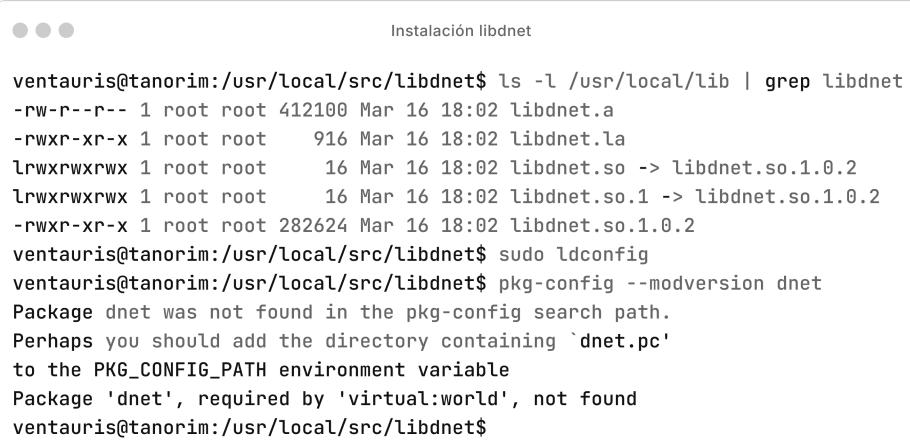
Ahora usamos sudo make install para instalar libdnet en el sistema. Este comando copia los archivos compilados a sus directorios correspondientes, asegurando que puedan ser utilizados por otras aplicaciones y librerías.



```
ventauris@tanorim:/usr/local/src/libdnet$ sudo make install
Making install in include
make[1]: Entering directory '/usr/local/src/libdnet/include'
Making install in dnet
make[2]: Entering directory '/usr/local/src/libdnet/include/dnet'
make[3]: Entering directory '/usr/local/src/libdnet/include/dnet'
make[3]: Nothing to be done for 'install-exec-am'.
/usr/bin/mkdir -p '/usr/local/include/dnet'
/usr/bin/install -c -m 644 addr.h arp.h blob.h eth.h fw.h icmp.h intf.h ip.h ip6.h
.
.
.
make[1]: Leaving directory '/usr/local/src/libdnet'
ventauris@tanorim:/usr/local/src/libdnet$
```

Figura 6.5: Instalando libdnet en el sistema.

Listamos los archivos de libdnet en /usr/local/lib para asegurarnos de que se instalaron correctamente. Luego, intentamos verificar su versión con pkg-config --modversion dnet, pero aparece un error indicando que no se encuentra en el PKG\_CONFIG\_PATH. Esto indica que es necesario añadir la ruta correcta para que el sistema reconozca la librería.



```
ventauris@tanorim:/usr/local/src/libdnet$ ls -l /usr/local/lib | grep libdnet
-rw-r--r-- 1 root root 412100 Mar 16 18:02 libdnet.a
-rwxr-xr-x 1 root root    916 Mar 16 18:02 libdnet.la
lrwxrwxrwx 1 root root      16 Mar 16 18:02 libdnet.so -> libdnet.so.1.0.2
lrwxrwxrwx 1 root root      16 Mar 16 18:02 libdnet.so.1 -> libdnet.so.1.0.2
-rwxr-xr-x 1 root root 282624 Mar 16 18:02 libdnet.so.1.0.2
ventauris@tanorim:/usr/local/src/libdnet$ sudo ldconfig
ventauris@tanorim:/usr/local/src/libdnet$ pkg-config --modversion dnet
Package dnet was not found in the pkg-config search path.
Perhaps you should add the directory containing `dnet.pc'
to the PKG_CONFIG_PATH environment variable
Package 'dnet', required by 'virtual:world', not found
ventauris@tanorim:/usr/local/src/libdnet$
```

Figura 6.6: Verificación de la instalación de libdnet.

Hemos editado el archivo dnet.pc usando nano, añadiendo las rutas correctas para que pkg-config pueda encontrar libdnet. Definimos las variables libdir e includedir, asegurándonos de que apunten a /usr/local/lib y /usr/local/include, respectivamente. Esto soluciona el problema anterior donde pkg-config no encontraba la librería.

```

GNU nano 7.2
prefix=/usr/local
exec_prefix=${prefix}
libdir=${exec_prefix}/lib
includedir=${prefix}/include

Name: dnet
Description: libdnet
Version: 1.0
Libs: -L${libdir} -ldnet
Cflags: -I${includedir}

```

Figura 6.7: Configurando libdnet para que sea reconocido por el sistema.

Ahora configuraremos el entorno para que pkg-config pueda encontrar libdnet. Primeiro, exportamos la variable PKG\_CONFIG\_PATH con la ruta correcta y la agregamos permanentemente al archivo /.bashrc. Luego, recargamos la configuración con source /.bashrc, actualizamos las librerías con ldconfig y verificamos que libdnet es reconocido correctamente ejecutando pkg-config --modversion dnet, confirmando la versión instalada.

```

Instalación libdnet

ventauris@tanorim:/usr/local/src/libdnet$ export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:$PKG_CONFIG_PATH
ventauris@tanorim:/usr/local/src/libdnet$ echo 'export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:$PKG_CONFIG_PATH' >> ~/.bashrc
ventauris@tanorim:/usr/local/src/libdnet$ source ~/.bashrc
ventauris@tanorim:/usr/local/src/libdnet$ sudo ldconfig
ventauris@tanorim:/usr/local/src/libdnet$ pkg-config --modversion dnet
1.0
ventauris@tanorim:/usr/local/src/libdnet$
```

Figura 6.8: Añadiendo libdnet al PATH y verificando su reconocimiento.

Clonamos el repositorio de libdaq, un componente necesario para Snort 3, desde su fuente oficial en GitHub. Este paso descarga el código fuente necesario para su compilación e instalación en la Raspberry Pi.

```
● ● ● Instalación libdaq

ventauris@tanorim:/usr/local/src$ sudo git clone https://github.com/snort3/libdaq.git
Cloning into 'libdaq'...
remote: Enumerating objects: 2584, done.
remote: Counting objects: 100% (177/177), done.
remote: Compressing objects: 100% (72/72), done.
remote: Total 2584 (delta 126), reused 117 (delta 105), pack-reused 2407 (from 2)
Receiving objects: 100% (2584/2584), 1.28 MiB | 4.43 MiB/s, done.
Resolving deltas: 100% (1834/1834), done.
ventauris@tanorim:/usr/local/src$
```

Figura 6.9: Descargando libdaq desde el repositorio oficial.

Entramos en el directorio de libdaq y ejecutamos el script bootstrap, que se encarga de generar los archivos de configuración necesarios para compilar el software correctamente. Se usa autoreconf para asegurarse de que todos los scripts de compilación estén en orden.

```
● ● ● Instalación libdaq

ventauris@tanorim:/usr/local/src$ cd libdaq
ventauris@tanorim:/usr/local/src/libdaq$ sudo ./bootstrap
+ autoreconf -ivf --warnings=all
.
.
.

autoreconf: Leaving directory '.'
ventauris@tanorim:/usr/local/src/libdaq$
```

Figura 6.10: Preparando libdaq para su compilación.

Ejecutamos ./configure para preparar el entorno de compilación de libdaq. Este script revisa que el sistema tenga todas las dependencias necesarias y genera los archivos de configuración adecuados para compilar el software sin problemas.



A terminal window titled "Instalación libdaq" showing the execution of the ./configure command. The terminal prompt is "ventauris@tanorim:/usr/local/src/libdaq\$". The command "sudo ./configure" is run, followed by "checking for a BSD-compatible install... /usr/bin/install -c". Three dots indicate the command is still running.

```
ventauris@tanorim:/usr/local/src/libdaq$ sudo ./configure
checking for a BSD-compatible install... /usr/bin/install -c
.
.
.
ventauris@tanorim:/usr/local/src/libdaq$
```

Figura 6.11: Configurando libdaq antes de la compilación.

Ejecutamos make para compilar libdaq. Este proceso traduce el código fuente a binarios ejecutables, asegurándose de que todas las dependencias y archivos necesarios se generen correctamente.



A terminal window titled "Instalación libdaq" showing the execution of the make command. The terminal prompt is "ventauris@tanorim:/usr/local/src/libdaq\$". The command "sudo make" is run, followed by "make all-recursive", "make[1]: Entering directory '/usr/local/src/libdaq'", "Making all in api", "make[2]: Entering directory '/usr/local/src/libdaq/api'", three dots indicating the process is still running, and finally "ventauris@tanorim:/usr/local/src/libdaq\$".

```
ventauris@tanorim:/usr/local/src/libdaq$ sudo make
make all-recursive
make[1]: Entering directory '/usr/local/src/libdaq'
Making all in api
make[2]: Entering directory '/usr/local/src/libdaq/api'
.
.
.
ventauris@tanorim:/usr/local/src/libdaq$
```

Figura 6.12: Compilando libdaq.

Ejecutamos sudo make install para instalar libdaq en el sistema. Esto copia los archivos compilados a sus ubicaciones correspondientes para que puedan ser utilizados por Snort y otros programas que lo requieran.

```
● ● ●           Instalación libdaq

ventauris@tanorim:/usr/local/src/libdaq$ sudo make install
Making install in api
make[1]: Entering directory '/usr/local/src/libdaq/api'
make[2]: Entering directory '/usr/local/src/libdaq/api'
.
.
.
make[1]: Leaving directory '/usr/local/src/libdaq'
ventauris@tanorim:/usr/local/src/libdaq$
```

Figura 6.13: Instalando libdaq.

Listamos los archivos de configuración de libdaq en /usr/local/lib/pkgconfig/ para asegurarnos de que la instalación se haya completado correctamente. Después, exportamos la variable PKG\_CONFIG\_PATH para que el sistema reconozca la librería. Finalmente, usamos pkg-config --modversion libdaq para confirmar que la versión instalada es la 3.0.19.

```
● ● ●           Instalación libdaq

ventauris@tanorim:/usr/local/src/libdaq$ ls -l /usr/local/lib/pkgconfig | grep libdaq
-rw-r--r-- 1 root root 291 Mar 16 19:39 libdaq.pc
-rw-r--r-- 1 root root 342 Mar 16 19:39 libdaq_static_afpacket.pc
-rw-r--r-- 1 root root 322 Mar 16 19:39 libdaq_static_bpf.pc
-rw-r--r-- 1 root root 316 Mar 16 19:39 libdaq_static_dump.pc
-rw-r--r-- 1 root root 314 Mar 16 19:39 libdaq_static_fst.pc
-rw-r--r-- 1 root root 309 Mar 16 19:39 libdaq_static_gwlb.pc
-rw-r--r-- 1 root root 326 Mar 16 19:39 libdaq_static_pcapy.pc
-rw-r--r-- 1 root root 325 Mar 16 19:39 libdaq_static_savefile.pc
-rw-r--r-- 1 root root 313 Mar 16 19:39 libdaq_static_trace.pc
ventauris@tanorim:/usr/local/src/libdaq$ export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:
$PKG_CONFIG_PATH
ventauris@tanorim:/usr/local/src/libdaq$ pkg-config --modversion libdaq
3.0.19
ventauris@tanorim:/usr/local/src/libdaq$
```

Figura 6.14: Verificando la instalación de libdaq.

Instalamos libhwloc-dev, una librería necesaria para la ejecución de Snort 3 y sus dependencias. Se incluyen automáticamente otros paquetes adicionales como libhwloc-plugins, libnuma-dev y libpciaccess0, que ayudan en la gestión de hardware y optimización del rendimiento.

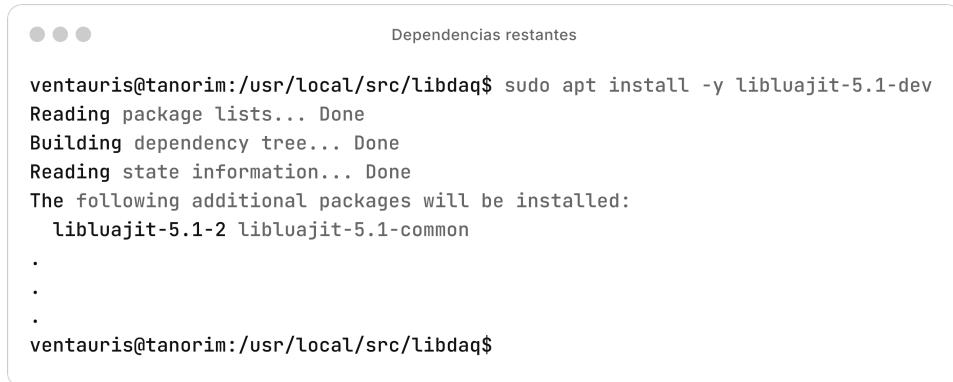


```
Dependencias restantes

ventauris@tanorim:/usr/local/src/libdaq$ sudo apt install -y libhwloc-dev
[sudo] password for ventauris:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
libhwloc-plugins libhwloc15 libnuma-dev libpciaccess0 libxnvctrl0 ocl-icd-libopencl1
.
.
.
ventauris@tanorim:/usr/local/src/libdaq$
```

Figura 6.15: Instalando dependencias necesarias.

Aquí instalamos libluajit-5.1-dev, más dependencias para Snort 3, ya que Snort utiliza LuaJIT para la configuración y personalización de reglas. También se instalan automáticamente libluajit-5.1-2 y libluajit-5.1-common, que contienen las bibliotecas compartidas necesarias para funcionar correctamente.

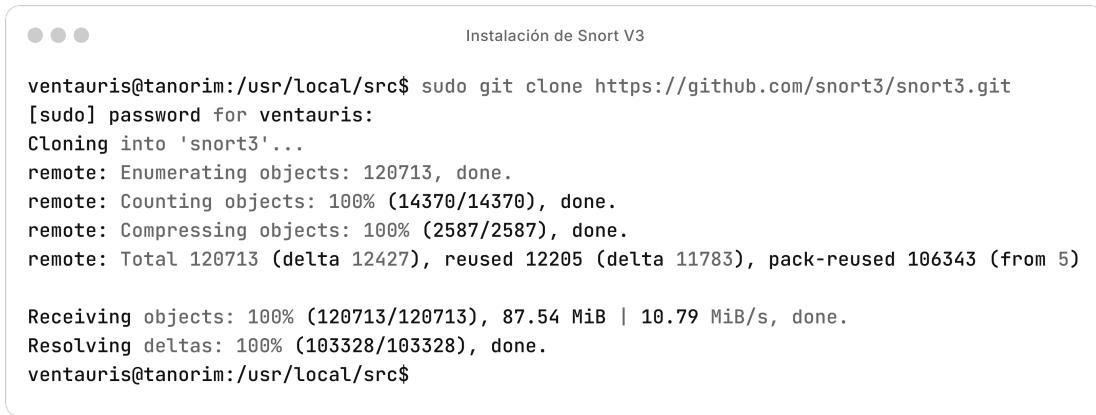


```
Dependencias restantes

ventauris@tanorim:/usr/local/src/libdaq$ sudo apt install -y libluajit-5.1-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
libluajit-5.1-2 libluajit-5.1-common
.
.
.
ventauris@tanorim:/usr/local/src/libdaq$
```

Figura 6.16: Instalando LuaJIT para Snort.

Descargamos el código fuente de Snort 3 directamente desde su repositorio oficial en GitHub mediante git clone.



```
Instalación de Snort V3

ventauris@tanorim:/usr/local/src$ sudo git clone https://github.com/snort3/snort3.git
[sudo] password for ventauris:
Cloning into 'snort3'...
remote: Enumerating objects: 120713, done.
remote: Counting objects: 100% (14370/14370), done.
remote: Compressing objects: 100% (2587/2587), done.
remote: Total 120713 (delta 12427), reused 12205 (delta 11783), pack-reused 106343 (from 5)

Receiving objects: 100% (120713/120713), 87.54 MiB | 10.79 MiB/s, done.
Resolving deltas: 100% (103328/103328), done.
ventauris@tanorim:/usr/local/src$
```

Figura 6.17: Clonando el repositorio de Snort 3.

Aquí establecemos la variable my\_path para definir el directorio de instalación de Snort. Luego, agregamos esta configuración al archivo /.bashrc para que se cargue automáticamente en futuras sesiones. Finalmente, usamos source /.bashrc para aplicar los cambios de inmediato.



```
Instalación de Snort V3

ventauris@tanorim:/usr/local/src/snort3$ export my_path=/usr/local/snort
ventauris@tanorim:/usr/local/src/snort3$ echo 'export my_path=/usr/local/snort' >>
~/bashrc
ventauris@tanorim:/usr/local/src/snort3$ source ~/bashrc
ventauris@tanorim:/usr/local/src/snort3$
```

Figura 6.18: Definiendo la ruta de instalación de Snort.

Instalamos Flex y Bison, dos herramientas para el análisis léxico y sintáctico en la compilación de Snort. Después de la instalación, verificamos la versión de Flex con flex --version para asegurarnos de que se instaló correctamente.

```
● ● ● Dependencias restantes

ventauris@tanorim:/usr/local/src/snort3$ sudo apt install flex bison -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libfl-dev libfl2
.
.
.
ventauris@tanorim:/usr/local/src/snort3$ flex --version
flex 2.6.4
```

Figura 6.19: Instalando Flex y Bison, herramientas necesarias para la compilación.

En este paso, instalamos libpcre2-dev, una biblioteca para el manejo de expresiones regulares en Snort 3. Nos es útil para la detección de patrones en el tráfico de red.

```
● ● ● Dependencias restantes

ventauris@tanorim:/usr/local/src/snort3$ sudo apt install libpcre2-dev -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libpcre2-16-0 libpcre2-32-0 libpcre2-posix3
.
.
.
ventauris@tanorim:/usr/local/src/snort3$
```

Figura 6.20: Instalando la biblioteca PCRE2 para el manejo de expresiones regulares.

Ejecutamos el script configure\_cmake.sh para configurar la compilación de Snort 3. Se especifica el prefijo de instalación con \$my\_path. Se generan los archivos de configuración y se definen las opciones de características, incluyendo los módulos DAQ que se activarán.

```
● ● ●
Instalación de Snort V3

ventauris@tanorim:/usr/local/src/snort3$ sudo ./configure_cmake.sh --prefix=$my_path
./configure_cmake.sh: 523: [: Illegal number:
Build Directory : build
Source Directory: /usr/local/src/snort3
CMake Warning:
  Ignoring empty string ("") provided on the command line.

CMake Deprecation Warning at CMakeLists.txt:1 (cmake_minimum_required):
  Compatibility with CMake < 3.5 will be removed from a future version of
  CMake.

  Update the VERSION argument <min> value or use a ...<max> suffix to tell
  CMake that the project does not need compatibility with older versions.

.

.

.

Feature options:
  DAQ Modules:      Static (afpacket;bpf;dump;fst;gwlb;pcap;savefile;trace)
  libatomic:         System-provided
  Hyperscan:        OFF
  ICONV:            ON
  Libunwind:        OFF
  LZMA:             ON
  RPC DB:           Built-in
  SafeC:            OFF
  TCMalloc:          OFF
  JEMalloc:          OFF
  UUID:              OFF
  NUMA:              ON
  LibML:             OFF
-----
-- Configuring done (5.3s)
-- Generating done (0.4s)
-- Build files have been written to: /usr/local/src/snort3/build
ventauris@tanorim:/usr/local/src/snort3$
```

Figura 6.21: Configurando Snort 3 con CMake.

Compilamos Snort con make -j \$(nproc), lo que nos permite aprovechar todos los núcleos del procesador para acelerar el proceso.



The image shows a terminal window titled "Instalación de Snort V3". The window has three gray dots at the top left. The terminal content displays the compilation process of Snort version 3. It shows numerous build steps for various CXX objects across different source directories, such as tools/snort2lua/preprocessor\_states/CMakeFiles/preprocessor\_states.dir/pps\_imap.cc.o, tools/snort2lua/preprocessor\_states/CMakeFiles/preprocessor\_states.dir/pps\_modbus.cc.o, and tools/snort2lua/preprocessor\_states/CMakeFiles/preprocessor\_states.dir/pps\_rna.cc.o. The progress bar indicates completion at 100% for most objects. The final output shows the linking of the executable snort, the building of the target preprocessor\_states, and the linking of the target snort2lua. The command ventauris@tanorim:/usr/local/src/snort3/build\$ is visible at the bottom.

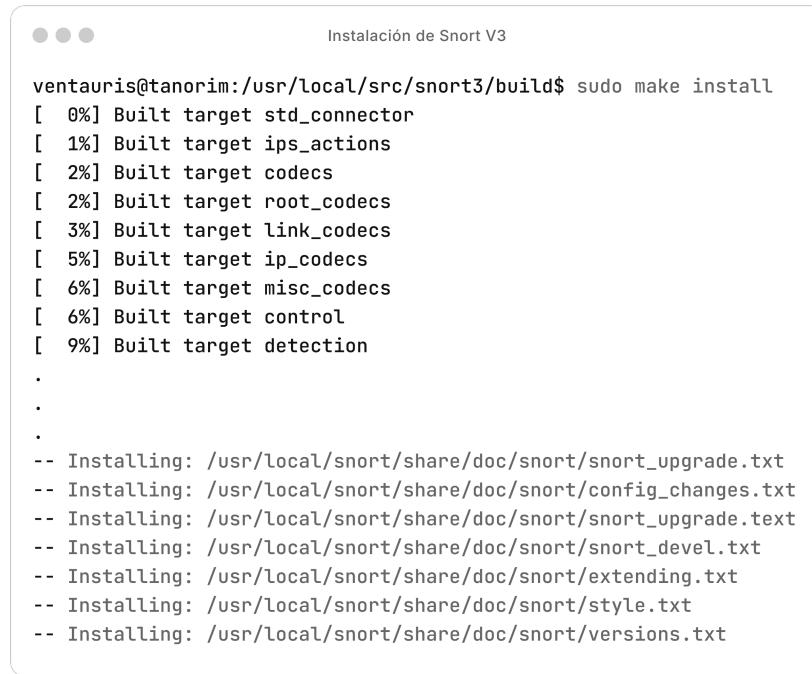
```
Instalación de Snort V3

.
.
.

[ 98%] Building CXX object tools/snort2lua/preprocessor_states/CMakeFiles/preprocessor_states.dir/pps_imap.cc.o
[ 98%] Building CXX object tools/snort2lua/preprocessor_states/CMakeFiles/preprocessor_states.dir/pps_modbus.cc.o
[ 98%] Building CXX object tools/snort2lua/preprocessor_states/CMakeFiles/preprocessor_states.dir/pps_rna.cc.o
[100%] Building CXX object tools/snort2lua/preprocessor_states/CMakeFiles/preprocessor_states.dir/pps_smtp.cc.o
[100%] Building CXX object tools/snort2lua/preprocessor_states/CMakeFiles/preprocessor_states.dir/pps_sfportscan.cc.o
[100%] Building CXX object tools/snort2lua/preprocessor_states/CMakeFiles/preprocessor_states.dir/pps_stream5_ip.cc.o
[100%] Building CXX object tools/snort2lua/preprocessor_states/CMakeFiles/preprocessor_states.dir/pps_stream5_global.cc.o
[100%] Linking CXX executable snort
[100%] Building CXX object tools/snort2lua/preprocessor_states/CMakeFiles/preprocessor_states.dir/pps_stream5_tcp.cc.o
[100%] Building CXX object tools/snort2lua/preprocessor_states/CMakeFiles/preprocessor_states.dir/pps_stream5_udp.cc.o
[100%] Building CXX object tools/snort2lua/preprocessor_states/CMakeFiles/preprocessor_states.dir/pps_stream5_ha.cc.o
[100%] Building CXX object tools/snort2lua/preprocessor_states/CMakeFiles/preprocessor_states.dir/preprocessor_api.cc.o
[100%] Built target preprocessor_states
[100%] Building CXX object tools/snort2lua/CMakeFiles/snort2lua.dir/snort2lua.cc.o
[100%] Building CXX object tools/snort2lua/CMakeFiles/snort2lua.dir/init_state.cc.o
[100%] Linking CXX executable snort2lua
[100%] Built target snort2lua
[100%] Built target snort
ventauris@tanorim:/usr/local/src/snort3/build$
```

Figura 6.22: Compilación de Snort.

Instalamos Snort tras haberlo compilado con sudo make install. Este comando copia los archivos generados en sus respectivas ubicaciones dentro del sistema.

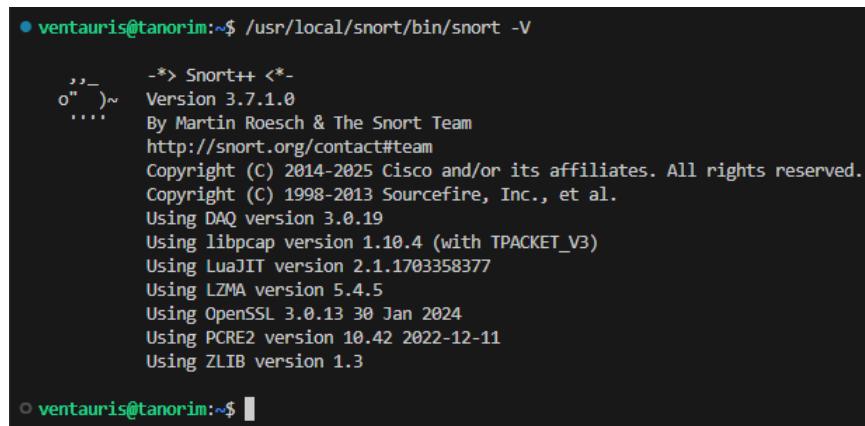


The terminal window title is "Instalación de Snort V3". The command entered is "sudo make install". The output shows the progress of building targets (std\_connector, ips\_actions, codecs, root\_codecs, link\_codecs, ip\_codecs, misc\_codecs, control, detection) and then installing various configuration and documentation files like snort\_upgrade.txt, config\_changes.txt, and style.txt.

```
ventauris@tanorim:/usr/local/src/snort3/build$ sudo make install
[  0%] Built target std_connector
[  1%] Built target ips_actions
[  2%] Built target codecs
[  2%] Built target root_codecs
[  3%] Built target link_codecs
[  5%] Built target ip_codecs
[  6%] Built target misc_codecs
[  6%] Built target control
[  9%] Built target detection
.
.
.
-- Installing: /usr/local/snort/share/doc/snort/snort_upgrade.txt
-- Installing: /usr/local/snort/share/doc/snort/config_changes.txt
-- Installing: /usr/local/snort/share/doc/snort/snort_upgrade.text
-- Installing: /usr/local/snort/share/doc/snort/snort-devel.txt
-- Installing: /usr/local/snort/share/doc/snort/extending.txt
-- Installing: /usr/local/snort/share/doc/snort/style.txt
-- Installing: /usr/local/snort/share/doc/snort/versions.txt
```

Figura 6.23: Instalación de Snort.

Finalmente verificamos que Snort se ha instalado correctamente con el comando /usr/local/snort/bin/snort -V. Esto nos muestra la versión instalada (3.7.1.0) junto con las bibliotecas y dependencias utilizadas, como DAQ, libpcap, LuaJIT, OpenSSL, entre otras.



The terminal window title is "ventauris@tanorim:~\$". The command entered is "/usr/local/snort/bin/snort -V". The output displays the Snort version (3.7.1.0), copyright information, and the versions of its dependencies: DAQ 3.0.19, libpcap 1.10.4, LuaJIT 2.1.1703358377, LZMA 5.4.5, OpenSSL 3.0.13, PCRE2 10.42, and ZLIB 1.3.

```
ventauris@tanorim:~$ /usr/local/snort/bin/snort -V
,,-      -*> Snort++ <*-.
o" )~ Version 3.7.1.0
     By Martin Roesch & The Snort Team
     http://snort.org/contact#team
     Copyright (C) 2014-2025 Cisco and/or its affiliates. All rights reserved.
     Copyright (C) 1998-2013 Sourcefire, Inc., et al.
     Using DAQ version 3.0.19
     Using libpcap version 1.10.4 (with TPACKET_V3)
     Using LuaJIT version 2.1.1703358377
     Using LZMA version 5.4.5
     Using OpenSSL 3.0.13 30 Jan 2024
     Using PCRE2 version 10.42 2022-12-11
     Using ZLIB version 1.3
```

Figura 6.24: Snort instalado con éxito.

Creamos el directorio de configuración de Snort (/usr/local/snort/etc/snort) y copiamos los archivos de configuración en formato Lua desde el directorio fuente de Snort 3. Luego, ejecutamos Snort con la configuración especificada para validar que todo esté correctamente configurado. La salida muestra que Snort ha cargado las reglas y módulos sin errores ni advertencias.

```
Instalación de Snort V3

ventauris@tanorim:~$ sudo mkdir -p /usr/local/snort/etc/snort
ventauris@tanorim:~$ sudo cp /usr/local/src/snort3/lua/*.lua /usr/local/snort/etc/snort/
ventauris@tanorim:~$ /usr/local/snort/bin/snort -c /usr/local/snort/etc/snort/snort.lua
-----
o")~  Snort++ 3.7.1.0
-----
Loading /usr/local/snort/etc/snort/snort.lua:
Loading snort_defaults.lua:
Finished snort_defaults.lua:
    stream_ip
    stream_icmp
    .
    .
    .
search engine (ac_bnfa)
    instances: 2
    patterns: 438
    pattern chars: 2602
    num states: 1832
    num match states: 392
    memory scale: KB
    total memory: 71.2812
    pattern memory: 19.6484
    match list memory: 28.4375
    transition memory: 22.9453
appid: MaxRss diff: 2944
appid: patterns loaded: 300
-----
pcap DAQ configured to passive.

Snort successfully validated the configuration (with 0 warnings).
o")~  Snort exiting
ventauris@tanorim:~$
```

Figura 6.25: Configuración de Snort validada con éxito.

### 6.3.2. Instalación de reglas y plugins

Tras la correcta instalación de Snort, nos aprovecharemos de la versión modular que nos trae la nueva versión 3, que maneja archivos con extensión .lua para manejar su configuración. A continuación se mostrará una guía paso a paso de como instalar las reglas básicas de la comunidad de Snort.

Descargamos las Community Rules de Snort con wget, descomprimiéndolas con tar.

```
● ● ● Descarga de las community rules

ventauris@tanorim:/usr/local/snort/etc/snort$ sudo wget https://www.snort.org/downloads/
community/snort3-community-rules.tar.gz
.
.
.

snort3-community-rules.tar.gz
100%[=====] 323.67K  998KB/s   in 0.3s

2025-03-21 22:58:21 (998 KB/s) - 'snort3-community-rules.tar.gz' saved [331442/331442]

ventauris@tanorim:/usr/local/snort/etc/snort$ sudo tar -xvzf snort3-community-rules.tar.gz
snort3-community-rules/
snort3-community-rules/snort3-community.rules
snort3-community-rules/VRT-License.txt
snort3-community-rules/LICENSE
snort3-community-rules/AUTHORS
snort3-community-rules/sid-msg.map
ventauris@tanorim:/usr/local/snort/etc/snort$ ls
AUTHORS balanced.lua      custom.rules      inline.lua      security.lua
snort.conf snort3-community-rules      snort_defaults.lua
LICENSE connectivity.lua    file_magic.rules  max_detect.lua  sensitive_data.rules
snort.lua  snort3-community-rules.tar.gz talos.lua
ventauris@tanorim:/usr/local/snort/etc/snort$ cd snort3-community-rules/
ventauris@tanorim:/usr/local/snort/etc/snort/snort3-community-rules$ ls
AUTHORS LICENSE VRT-License.txt sid-msg.map snort3-community.rules
ventauris@tanorim:/usr/local/snort/etc/snort/snort3-community-rules$
```

Figura 6.26: Añadiendo reglas preconfiguradas a Snort.

Posteriormente editamos el archivo snort.lua, ubicado en /usr/local/snort/etc/snort/, para incluir las reglas de community.rules. Esto permite que Snort cargue estas reglas al iniciar y pueda detectar amenazas basadas en ellas. La edición se puede hacer con nano, vim o cualquier otro editor de texto.

```
ips =
{
    -- use this to enable decoder and inspector alerts
    --enable_builtin_rules = true,

    -- use include for rules files; be sure to set your path
    -- note that rules files can include other rules files
    -- (see also related path vars at the top of snort_defaults.lua)
    rules = [[
        include /usr/local/snort/etc/snort/snort3-community-rules/snort3-community.rules
    ]],
    variables = default_variables
}
```

Figura 6.27: Modificación de snort.lua para agregar las reglas preconfiguradas.

Una vez agregadas las reglas a snort.lua, hacemos la prueba para comprobar que no ha habido ningún error.



The terminal window shows the command `sudo snort -c /usr/local/snort/etc/snort/snort.lua` being run. The output indicates the configuration was successfully validated with 0 warnings, and Snort is exiting.

```
● ● ● Comprobación de las reglas
ventauris@tanorim:/usr/local/snort/etc/snort$ sudo snort -c /usr/local/snort/etc/snort/
snort.lua
-----
o")~  Snort++ 3.7.1.0
-----
Loading /usr/local/snort/etc/snort/snort.lua:
Loading snort_defaults.lua:
Finished snort_defaults.lua:
.
.
.

pcap DAQ configured to passive.

Snort successfully validated the configuration (with 0 warnings).
o")~  Snort exiting
ventauris@tanorim:/usr/local/snort/etc/snort$
```

Figura 6.28: Validación configuración de Snort.

### 6.3.3. Configuración preprocesador HTTP Inspect

Accedemos a snort.lua y buscamos http\_inspect = y escribimos las reglas adecuadas para la gestión de una PYME. Las reglas incluyen:

Parámetro	Valor	Uso en una PYME
request_depth	-1	Inspeccionar todo el contenido de la petición (detección de inyecciones o malware en el cuerpo).
response_depth	-1	Revisar la totalidad de la respuesta enviada por el servidor.
unzip	true	Descomprimir gzip/deflate (evita que contenido malicioso comprimido pase desapercibido).
oversize_dir_length	500	Alertar si una ruta de la URI supera longitud excesiva (potenciales intentos de ataque).
maximum_headers	200	Detectar exceso de cabeceras (protege ante ataques por cabeceras anómalas).

Cuadro 6.1: Parámetros de http\_inspect.

```
-- http_inspect para inspección HTTP
http_inspect =
{
    -- Escanear todo el cuerpo de la petición/respuesta (ojo a la carga en una Pi)
    request_depth = -1,
    response_depth = -1,

    -- Activa descompresión de gzip/deflate para inspeccionar payload
    unzip = true,

    -- Longitud máxima de directorio en URI, pasado este valor se dispara alerta 119:15
    oversize_dir_length = 500,

    -- Número máximo de cabeceras permitidas (ej. 200), si se superan -> alerta 119:20
    maximum_headers = 200,

    -- Tamaño máximo (en bytes) de una cabecera individual antes de alertar 119:19
    maximum_header_length = 4096,

    -- ¿Normalizar caracteres UTF en las respuestas?
    normalize_utf = true,

    -- Descomprimir PDF, SWF, ZIP, etc. (cuidado con rendimiento)
    decompress_pdf = false,
    decompress_swf = false,
    decompress_zip = false,
    decompress_vba = false,

    -- Profundidad de escaneo en adjuntos MIME
    max_mime_attach = 5,

    -- Ejemplo: bloquear (o alertar) si el cliente usa ciertos métodos
    --allowed_methods = 'GET,POST,HEAD,OPTIONS',
    --disallowed_methods = 'DELETE,TRACE,TRACK'
    -- (Solo activalo si estás seguro de que tu app no requiere esos métodos)

    -- Manejo de + como espacio en URIs
    plus_to_space = true
}
```

Figura 6.29: Configuración http\_inspect.

Además de la configuración se necesita agregar al final de la sección de binders en snort.lua el siguiente bloque de código para el correcto funcionamiento.

```

binder =
{
    -- port bindings required for protocols without wizard support
    { when = { proto = 'udp', ports = '53', role='server' }, use = { type = 'dns' } },
    { when = { proto = 'tcp', ports = '53', role='server' }, use = { type = 'dns' } },
    { when = { proto = 'tcp', ports = '111', role='server' }, use = { type = 'rpc_decode' } },
    { when = { proto = 'tcp', ports = '502', role='server' }, use = { type = 'modbus' } },
    { when = { proto = 'tcp', ports = '2123 2152 3386', role='server' }, use = { type = 'gtp_inspect' } },
    { when = { proto = 'tcp', ports = '2404', role='server' }, use = { type = 'iec104' } },
    { when = { proto = 'udp', ports = '2222', role = 'server' }, use = { type = 'cip' } },
    { when = { proto = 'tcp', ports = '44818', role = 'server' }, use = { type = 'cip' } },

    { when = { proto = 'tcp', service = 'dcerpc' }, use = { type = 'dce_tcp' } },
    { when = { proto = 'udp', service = 'dcerpc' }, use = { type = 'dce_udp' } },
    { when = { proto = 'udp', service = 'netflow' }, use = { type = 'netflow' } },

    { when = { service = 'netbios-ssn' }, use = { type = 'dce_smb' } },
    { when = { service = 'dce_http_server' }, use = { type = 'dce_http_server' } },
    { when = { service = 'dce_http_proxy' }, use = { type = 'dce_http_proxy' } },

    { when = { service = 'cip' }, use = { type = 'cip' } },
    { when = { service = 'dnsp3' }, use = { type = 'dns' } },
    { when = { service = 'dns' }, use = { type = 'dns' } },
    { when = { service = 'ftp' }, use = { type = 'ftp_server' } },
    { when = { service = 'ftp-data' }, use = { type = 'ftp_data' } },
    { when = { service = 'gtp' }, use = { type = 'gtp_inspect' } },
    { when = { service = 'imap' }, use = { type = 'imap' } },
    { when = { service = 'http' }, use = { type = 'http_inspect' } },
    { when = { service = 'http2' }, use = { type = 'http2_inspect' } },
    { when = { service = 'iec104' }, use = { type = 'iec104' } },
    { when = { service = 'mms' }, use = { type = 'mms' } },
    { when = { service = 'modbus' }, use = { type = 'modbus' } },
    { when = { service = 'pop3' }, use = { type = 'pop' } },
    { when = { service = 'ssh' }, use = { type = 'ssh' } },
    { when = { service = 'sip' }, use = { type = 'sip' } },
    { when = { service = 'smtp' }, use = { type = 'smtp' } },
    { when = { service = 'ssl' }, use = { type = 'ssl' } },
    { when = { service = 'sunrpc' }, use = { type = 'rpc_decode' } },
    { when = { service = 's7complus' }, use = { type = 's7complus' } },
    { when = { service = 'telnet' }, use = { type = 'telnet' } },

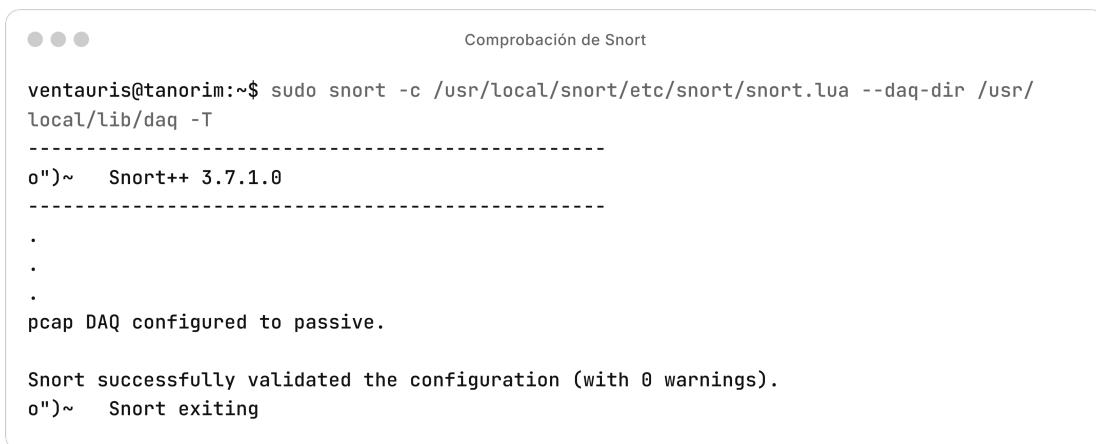
    { use = { type = 'wizard' } },
}

{
    when = { proto = 'tcp', ports = '80 443', role = 'server' },
    use = { type = 'http_inspect' }
}
}

```

Figura 6.30: Configuración de binders.

Por último validamos la configuración para asegurarnos de que no hay errores.



```

Comprobación de Snort

ventauris@tanorim:~$ sudo snort -c /usr/local/snort/etc/snort/snort.lua --daq-dir /usr/local/lib/daq -T
-----
o")~  Snort++ 3.7.1.0
-----
.
.
.

pcap DAQ configured to passive.

Snort successfully validated the configuration (with 0 warnings).
o")~  Snort exiting

```

Figura 6.31: Validación de la configuración.

### 6.3.4. SSL Inspector

El siguiente módulo que habilitaremos será el SSL Inspector, el procedimiento es similar al anterior. Comenzaremos por modificar el archivo snort.lua, buscamos el apartado de ssl y agregaremos lo siguiente.

```
ssl = {
    -- Por defecto, no se confía en servidores externos automáticamente
    trust_servers = false,
    -- Establece un límite para evitar ataques tipo Heartbleed
    max_heartbeat_length = 2048,
}
```

Figura 6.32: Configuración de SSL.

Posteriormente vincularemos la configuración ssl en la sección binder en caso de no tenerlo.

```
{ when = { service = 'ssl' }, use = { type = 'ssl' } }
```

Figura 6.33: Víncular con binders.

A continuación, podríamos verificar la configuración de snort, tras asegurarnos que la sintaxis es válida, reiniciamos snort mediante systemctl.

### 6.3.5. Stream IP

Antiguamente conocido como Frag3, ha sido actualizado a una nueva versión de nombre Stream IP. Su configuración será la siguiente en el archivo snort.lua.

```
stream_ip = {
    max_frags = 8192,          -- máximo número de fragmentos simultáneos
    max_overlaps = 5,           -- máximo número permitido de solapamientos (0 para ilimitado)
    min_frag_length = 128,      -- alerta si la longitud del fragmento es menor que 128 bytes
    min_ttl = 5,                -- ignora fragmentos con TTL menor a 5
    policy = 'linux',           -- política de reensamblado (por defecto recomendado)
    session_timeout = 60,        -- tiempo en segundos antes de eliminar una sesión de reensamblado IP
}
```

Figura 6.34: Configuración de Stream IP.

Validamos la sintaxis y reiniciamos snort de nuevo.

### 6.3.6. Stream TCP

Otro procesador que se conocía por el nombre de Stream5, su nombre ha cambiado a Stream TCP. Se configura de manera similar a Stream IP. Una vez hechos los cambios en snort.lua, validamos la sintaxis y reiniciamos Snort.

```
stream_tcp = {
    policy = 'linux',
    max_window = 1048576,
    overlap_limit = 10,
    max_pdu = 16384,
    reassemble_async = true,
    queue_limit = {
        max_bytes = 4194304,
        max_segments = 2048,
        asymmetric_ids_flush_threshold = 2097152, -- Umbral de descarga para flujos asimétricos (2 MB, protege memoria)
    },
    small_segments = {
        count = 5,
        maximum_size = 64,
    },
    session_timeout = 180,
    embryonic_timeout = 30,
    idle_timeout = 1800,
}
```

Figura 6.35: Configuración de Stream TCP.

### 6.3.7. Reputation

Este procesador trabaja bloqueando IPs sospechosas o que han sido documentadas como maliciosas usando listas. Para su configuración se ha creado una carpeta /reputation donde se guardará una lista de IPs sospechosas, las IPs se han obtenido de emergingthreats.net. Como ya es costumbre, tras realizar la configuración se valida la sintaxis de snort y se reinicia el servicio.

```
reputation = {
    blocklist = 'blocklist.rules',
    -- allowlist no es obligatoria ahora, pero se pueden hacer excepciones
    list_dir = '/usr/local/snort/etc/snort/reputation',
    memcap = 500,
    nested_ip = 'inner',
    priority = 'allowlist',
    scan_local = false,
    allow = 'do_not_block',
}
```

Figura 6.36: Configuración de Reputation.

### 6.3.8. Datos sensibles

En versiones anteriores de Snort existe el preprocesador Sensitive Data, sin embargo en las versiones más nuevas este se ha desechado, sin embargo vamos a adaptar algunas expresiones regulares para alertar si se comparten datos sensibles básicos de una PYME.

Creamos un nuevo archivo de reglas de nombre custom.rules y agregamos las siguientes reglas.

```
GNU nano 7.2                                         /usr/local/snort/etc/snort/custom.rules *
# Detección de emails en texto plano
alert tcp $HOME_NET any -> $EXTERNAL_NET any (
    msg:"Sensitive Data - Email detected";
    flow:established,to_server;
    pcre:'/[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}/';
    classtype:sdf; sid:1000001; rev:1;
)

# Detección de número de tarjeta de crédito Visa, Mastercard, Amex, Discover (16 dígitos básico)
alert tcp $HOME_NET any -> $EXTERNAL_NET any (
    msg:"Sensitive Data - Credit Card detected";
    flow:established,to_server;
    pcre:'/\b(?:4[0-9]{12}(?:[0-9]{3})?|5[1-5][0-9]{14}|3[47][0-9]{13}|6(?:011|5[0-9]{2})[0-9]{12})\b/';
    classtype:sdf; sid:1000002; rev:1;
)

#Detección de Número de Seguridad Social (NUSS) de España (formato: 12 dígitos, sin espacios)
alert tcp $HOME_NET any -> $EXTERNAL_NET any (
    msg:"Sensitive Data - NUSS España detectado";
    flow:established,to_server;
    pcre:'/\b\d{12}\b/';
    classtype:policy-violation; sid:1000003; rev:1;
)
```

Figura 6.37: Expresiones regulares para protección de datos sensibles.

Agregamos la ruta donde se encuentran las expresiones regulares en snort.lua, comprobamos de nuevo la sintaxis de snort mediante libdaq y reiniciamos el servicio.

```
ips =
{
    -- use this to enable decoder and inspector alerts
    --enable_builtin_rules = true,

    -- use include for rules files; be sure to set your path
    -- note that rules files can include other rules files
    -- (see also related path vars at the top of snort_defaults.lua)
    rules = [[
        include /usr/local/snort/etc/snort/snort3-community-rules/snort3-community.rules
        include /usr/local/snort/etc/snort/custom.rules
    ]],
    variables = default_variables
}
```

Figura 6.38: Agregación de custom.rules.

### 6.3.9. Antivirus ClamAV

Finalmente, la instalación del antivirus Clam AV.

```
● ventauris@tanorim:~$ sudo apt install clamav clamav-daemon -y
[sudo] password for ventauris:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  clamav-base clamav-freshclam clamdscan libclamav11t64
Suggested packages:
  libclamunrar clamav-docs daemon libclamunrar11
The following NEW packages will be installed:
  clamav clamav-base clamav-daemon clamav-freshclam clamdscan libclamav11t64
0 upgraded, 6 newly installed, 0 to remove and 16 not upgraded.
Need to get 9495 kB of archives.
After this operation, 38.3 MB of additional disk space will be used.
Get:1 http://ports.ubuntu.com/ubuntu-ports noble-updates/main arm64 clamav-base all 1.0.8+dfsg-0ubuntu0.24.04.1 [93.5 kB]
Get:2 http://ports.ubuntu.com/ubuntu-ports noble-updates/main arm64 libclamav11t64 arm64 1.0.8+dfsg-0ubuntu0.24.04.1 [4613 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports noble-updates/main arm64 clamav-freshclam arm64 1.0.8+dfsg-0ubuntu0.24.04.1 [96.7 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports noble-updates/main arm64 clamav-daemon arm64 1.0.8+dfsg-0ubuntu0.24.04.1 [211 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports noble-updates/main arm64 clamav arm64 1.0.8+dfsg-0ubuntu0.24.04.1 [4429 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports noble-updates/main arm64 clamdscan arm64 1.0.8+dfsg-0ubuntu0.24.04.1 [51.1 kB]
Fetched 9495 kB in 1s (7522 kB/s)
Preconfiguring packages ...
Selecting previously unselected package clamav-base.
(Reading database ... 88716 files and directories currently installed.)
Preparing to unpack .../0-clamav-base_1.0.8+dfsg-0ubuntu0.24.04.1_all.deb ...
Unpacking clamav-base (1.0.8+dfsg-0ubuntu0.24.04.1) ...
Selecting previously unselected package libclamav11t64:arm64.
Preparing to unpack .../1-libclamav11t64_1.0.8+dfsg-0ubuntu0.24.04.1_arm64.deb ...
Unpacking libclamav11t64:arm64 (1.0.8+dfsg-0ubuntu0.24.04.1) ...
Selecting previously unselected package clamav-freshclam.
Preparing to unpack .../2-clamav-freshclam_1.0.8+dfsg-0ubuntu0.24.04.1_arm64.deb ...
Unpacking clamav-freshclam (1.0.8+dfsg-0ubuntu0.24.04.1) ...
Selecting previously unselected package clamav-daemon.
Preparing to unpack .../3-clamav-daemon_1.0.8+dfsg-0ubuntu0.24.04.1_arm64.deb ...
Unpacking clamav-daemon (1.0.8+dfsg-0ubuntu0.24.04.1) ...
Selecting previously unselected package clamav.
Preparing to unpack .../4-clamav_1.0.8+dfsg-0ubuntu0.24.04.1_arm64.deb ...
Unpacking clamav (1.0.8+dfsg-0ubuntu0.24.04.1) ...
Selecting previously unselected package clamdscan.
Preparing to unpack .../5-clamdscan_1.0.8+dfsg-0ubuntu0.24.04.1_arm64.deb ...
Unpacking clamdscan (1.0.8+dfsg-0ubuntu0.24.04.1) ...
Setting up libclamav11t64:arm64 (1.0.8+dfsg-0ubuntu0.24.04.1) ...
Setting up clamav-base (1.0.8+dfsg-0ubuntu0.24.04.1) ...
Setting up clamav-freshclam (1.0.8+dfsg-0ubuntu0.24.04.1) ...
Setting up clamdscan (1.0.8+dfsg-0ubuntu0.24.04.1) ...
Setting up clamav-daemon (1.0.8+dfsg-0ubuntu0.24.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/clamav-daemon.service → /usr/lib/systemd/system/clamav-daemon.service.
Created symlink /etc/systemd/system/sockets.target.wants/clamav-daemon.socket → /usr/lib/systemd/system/clamav-daemon.socket.
```

Figura 6.39: Instalación ClamAV.

Detenemos el servicio de ClamAV actualizamos las firmas y volvemos a iniciar el servicio.

```
● ventauris@tanorim:~$ sudo systemctl stop clamav-freshclam
● ventauris@tanorim:~$ sudo freshclam
ClamAV update process started at Sun Mar 23 21:05:49 2025
Sun Mar 23 21:05:49 2025 -> daily.cvd database is up-to-date (version: 27586, sigs: 2074246, f-level: 90, builder: raynman)
Sun Mar 23 21:05:49 2025 -> main.cvd database is up-to-date (version: 62, sigs: 6647427, f-level: 90, builder: sigmgr)
Sun Mar 23 21:05:49 2025 -> bytecode.cvd database is up-to-date (version: 335, sigs: 86, f-level: 90, builder: raynman)
● ventauris@tanorim:~$ sudo systemctl start clamav-freshclam
```

Figura 6.40: Instalación ClamAV.

## **6.4. Generación de un script para la instalación automática**

A la espera del visto bueno, dudas y econtrar sustituo a fallas

## **7. Casos prácticos: utilización de R-Snort**

### **7.1. Entorno de trabajo**

### **7.2. Instalación**

Incluye capturas de pantalla relevantes.

### **7.3. Utilización y pruebas**

#### **7.3.1. Benchmark de rendimiento**

#### **7.3.2. Pruebas**

### **7.4. Resumen**

# **Resultados y discusión**

# Conclusiones

# Bibliografía

- [1] Snort Official Website. <https://www.snort.org/>
- [2] Raspberry Pi Official Website. <https://www.raspberrypi.org/>

## **A. Anexo A**

Manuales de instrucciones para los usuarios

## **B. Anexo B**

Aquí irán las capturas reales, puesto que se ha usado una página embellecedora para la captura de pantalla.