Suricata: Guía Completa de Instalación, Configuración, Pruebas, Administración y Ejemplos (con Reglas Predeterminadas)

¿Qué es Suricata?

Suricata es un sistema de detección y prevención de intrusiones (IDS/IPS) de código abierto. Monitorea el tráfico de red en tiempo real, lo analiza en busca de patrones sospechosos y puede tomar medidas para bloquear actividades maliciosas.

Funcionamiento de Suricata (Resumen por Puntos):

- Captura de Tráfico: Suricata "escucha" el tráfico de red que pasa por una interfaz específica.
- Decodificación: Descompone los paquetes de datos en elementos comprensibles (direcciones IP, puertos, protocolos, etc.).
- Inspección Profunda de Paquetes (DPI): Analiza el contenido de los paquetes para identificar aplicaciones, comandos y posibles datos maliciosos.
- Comparación con Reglas: Compara el tráfico con reglas predefinidas que describen patrones de actividad maliciosa o sospechosa.
- Generación de Alertas: Si el tráfico coincide con una regla, Suricata genera una alerta con detalles del evento.
- Registro: Guarda un registro detallado de todos los eventos, incluyendo alertas, para análisis posterior.
- Prevención (Opcional): Si se configura como IPS, Suricata puede bloquear el tráfico malicioso en tiempo real.

Listas de Reglas:

Las reglas son el núcleo de Suricata. Son instrucciones que definen qué tráfico es sospechoso. Suricata viene preconfigurado para usar las reglas de Emerging Threats Open (ET Open), pero debes descargarlas.

Instalación y Configuración:

Preparación del Sistema:
Objetivo: Asegurarse de que el sistema esté actualizado y tenga los repositorios necesarios para instalar Suricata.
Comandos:

- sudo apt update
- sudo apt upgrade

Instalación de Suricata:

Objetivo: Descargar e instalar el software Suricata y sus componentes necesarios.
Debian y Kali Linux:
sudo apt install suricata

En Ubuntu:

- sudo add-apt-repository ppa:oisf/suricata-stable
- sudo apt update
- sudo apt install suricata

Descarga de Reglas:

Objetivo: Obtener las reglas de detección de ET Open, que son esenciales para que Suricata funcione correctamente.
Comando:

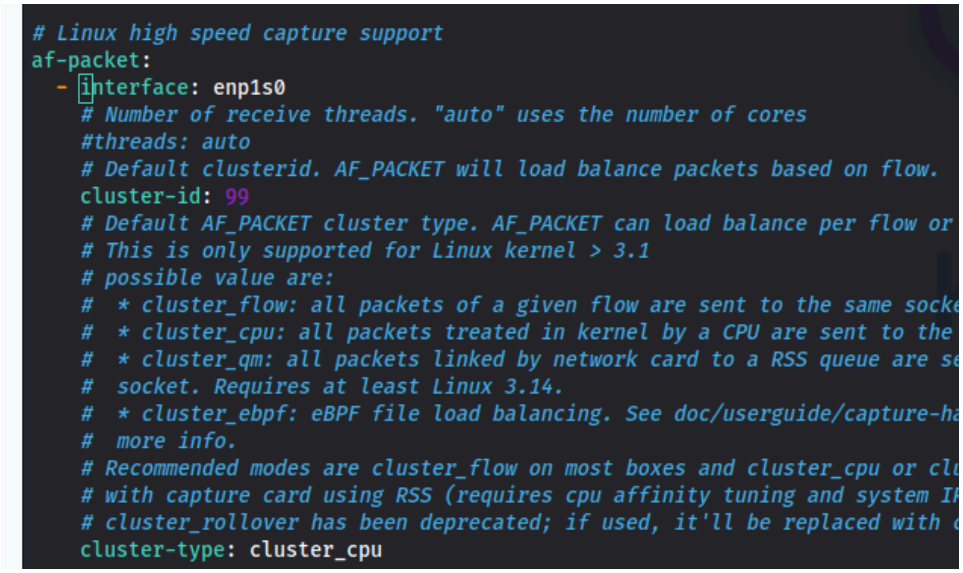
- sudo suricata-update

Configuración:

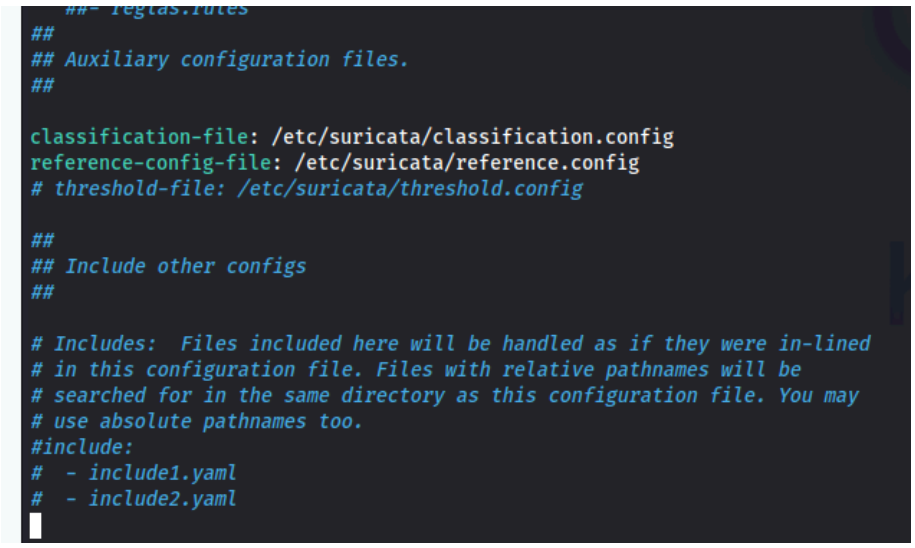
Objetivo: Personalizar el comportamiento de Suricata según tus necesidades y entorno de red. Puedes cambiar la interfaz de red y las rutas de las reglas.

Cómo: Edita el archivo de configuración suricata.yaml (ubicado en /etc/suricata/suricata.yaml).

Ejemplo de Configuración



```
# Linux high speed capture support
af-packet:
  - interface: enp1s0
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
    cluster-id: 99
    # Default AF_PACKET cluster type. AF_PACKET can load balance per flow or
    # This is only supported for Linux kernel > 3.1
    # possible value are:
    #  * cluster_flow: all packets of a given flow are sent to the same socke
    #  * cluster_cpu: all packets treated in kernel by a CPU are sent to the
    #  * cluster_qm: all packets linked by network card to a RSS queue are se
    #  socket. Requires at least Linux 3.14.
    #  * cluster_ebpf: eBPF file load balancing. See doc/userguide/capture-ha
    #  more info.
    # Recommended modes are cluster_flow on most boxes and cluster_cpu or clu
    # with capture card using RSS (requires cpu affinity tuning and system IR
    # cluster_rollover has been deprecated; if used, it'll be replaced with c
    cluster-type: cluster_cpu
```



```
##- reglas.rules
##
## Auxiliary configuration files.
##

classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config
# threshold-file: /etc/suricata/threshold.config

##
## Include other configs
##

# Includes:  Files included here will be handled as if they were in-lined
# in this configuration file. Files with relative pathnames will be
# searched for in the same directory as this configuration file. You may
# use absolute pathnames too.
#include:
#  - include1.yaml
#  - include2.yaml
```

Validación de la Configuración:

Objetivo: Asegurarse de que el archivo de configuración no tenga errores de sintaxis que puedan impedir que Suricata funcione correctamente.
Comando:

- suricata -T -c <config>: **Valida la configuración de Suricata.**
- suricata -c <config> -i <interfaz>: **Inicia Suricata en modo IDS en la interfaz especificada.**
- suricata -t: **Ejecuta Suricata en modo de prueba (para análisis de capturas de paquetes).**
- suricata -c <config> -s <reglas> -i <interfaz>: **Inicia Suricata con reglas personalizadas.**

Reemplaza:

- <config>: con la ruta al archivo de configuración (suricata.yaml).
- <reglas>: con la ruta al archivo de reglas personalizadas.
- <interfaz>: con el nombre de la interfaz de red a monitorear.

Ejemplo del comando “suricata -t”

```
root@luna-Inspiron-5570:/var/lib/suricata/rules# suricata -T
i: suricata: This is Suricata version 7.0.3 RELEASE running in SYSTEM mode
i: suricata: Configuration provided was successfully loaded. Exiting.
root@luna-Inspiron-5570:/var/lib/suricata/rules#
```

Inicio, Detención y Administración de Suricata:
Objetivo: Iniciar, detener y administrar el servicio de Suricata.
Comandos:

- sudo systemctl start suricata # Inicia Suricata.
- sudo systemctl stop suricata # Detiene a Suricata.
- sudo systemctl restart suricata # Reinicia Suricata.
- sudo systemctl status suricata # Verifica el estado de Suricata.
- sudo systemctl enable suricata # Habilita Suricata para que se inicie automáticamente al arrancar el sistema.
- sudo systemctl disable suricata # Deshabilita el inicio automático de Suricata.

Pruebas:

- Ataques Simulados: Utiliza herramientas como Metasploit o Nmap para simular ataques y verificar si Suricata los detecta.
- Generación de Tráfico Malicioso: (Con precaución) Genera tráfico que coincida con tus reglas para probar la detección.
- Análisis de Registros: Examina los archivos de registro de Suricata (ubicados en /var/log/suricata/) para verificar que los eventos se registran correctamente.

Análisis de Registros de Suricata:
Suricata, como sistema de detección de intrusiones (IDS), genera varios archivos de registro que proporcionan una visión detallada del tráfico de red y la actividad de seguridad. A continuación, se presenta un análisis de los principales archivos de registro y su relevancia en el contexto de un informe de seguridad:

1. eve.json:
Este archivo es el núcleo del registro de eventos de Suricata. Contiene una descripción exhaustiva de cada evento detectado, incluyendo:

1. Alertas de Seguridad: Detalles sobre posibles intrusiones, ataques o comportamientos anómalos en la red.
2. Flujos de Red: Información sobre las conexiones establecidas, incluyendo direcciones IP, puertos, protocolos y duración.
3. Registros DNS: Consultas y respuestas DNS, revelando qué dominios se están resolviendo.
4. Registros HTTP/TLS: Detalles de las solicitudes y respuestas HTTP, incluyendo encabezados, métodos y URLs, así como información sobre el tráfico TLS/SSL cifrado.

El archivo eve.json es fundamental para análisis forenses profundos, ya que permite reconstruir la actividad de red con gran detalle.

2. fast.log:
Este archivo actúa como un resumen ejecutivo de las alertas de seguridad. Cada línea representa una alerta, proporcionando información esencial como:

1. Marca de Tiempo: Cuándo se generó la alerta.
2. Fuente y Destino: Las direcciones IP y puertos involucrados en la conexión sospechosa.
3. Regla Activada: La regla específica de Suricata que desencadenó la alerta.
4. Clasificación: La categoría de la alerta (por ejemplo, ataque web, tráfico sospechoso).
5. Prioridad: La severidad de la alerta, ayudando a priorizar la investigación.

El archivo fast.log es ideal para una revisión rápida de las alertas más relevantes y urgentes.

3. stats.log:
Este archivo se centra en el rendimiento de Suricata, registrando métricas como:

1. Uso de CPU y Memoria: Para evaluar la carga del sistema y optimizar la configuración.
2. Paquetes Procesados: Para medir el volumen de tráfico analizado.
3. Reglas Coincidentes: Para identificar las reglas más activas y ajustar la detección.

El archivo stats.log es crucial para garantizar que Suricata funcione de manera eficiente y no afecte negativamente al rendimiento del sistema.

4. suricata.log:
Este archivo es el diario de Suricata, donde registra:

1. Mensajes de Inicio: Confirmando la versión y la configuración utilizada.
2. Errores y Advertencias: Señalando problemas potenciales o configuraciones incorrectas.
3. Notificaciones: Información sobre eventos importantes durante la ejecución.

El archivo suricata.log es esencial para la resolución de problemas y el mantenimiento del sistema.

Solución del error "fanout not supported by kernel" y optimización de Suricata

Al iniciar Suricata, el mensaje de error **"fanout not supported by kernel: Kernel too old or cluster-id 99 already in use"** indica que la función "fanout" de AF_PACKET, que distribuye la carga de procesamiento de paquetes entre núcleos de CPU, no puede ser utilizada.

Causas y soluciones:

Kernel incompatible:

- Causa: La versión del kernel de Linux es demasiado antigua y no soporta fanout.
- Solución: Actualiza tu kernel a una versión compatible. Consulta la documentación de tu distribución para obtener instrucciones específicas.

Cluster ID en uso:

- Causa: El ID de clúster que Suricata intenta usar (normalmente 99) ya está siendo utilizado por otro proceso en tu sistema.
- Solución:
- Cambia el Cluster ID: Modifica el valor de cluster-id en tu archivo suricata.yaml a un número no utilizado (por ejemplo, 100).
- Verifica otros procesos: Utiliza el comando ps aux | grep suricata para comprobar si hay otras instancias de Suricata ejecutándose que podrían estar usando el mismo ID.

Cluster Type Incorrecto:

- Causa: El tipo de clúster (cluster_type) seleccionado en la configuración no es compatible con tu sistema o no está configurado correctamente.
- Solución: Prueba diferentes tipos de clúster y asegúrate de que la configuración de afinidad de CPU e interrupciones sea correcta para los modos cluster_cpu y cluster_qm.

Modos de cluster:

1. **cluster_flow (Recomendado):** Agrupa paquetes por flujo de conexión (misma IP y puerto origen/destino). Mejora la localidad de datos y reduce la sobrecarga entre núcleos. Ideal para la mayoría de los sistemas.
2. **cluster_cpu:** Asigna paquetes secuencialmente a núcleos de CPU. Útil si tu tarjeta de red no soporta RSS o si cluster_flow no ofrece el

rendimiento deseado. Requiere ajustes de afinidad de CPU e interrupciones.

3. **cluster_qm:** Similar a cluster_cpu, pero con colas MPMC para mejorar la eficiencia. También requiere ajustes de afinidad de CPU e interrupciones.