Resumen Teórica 11 : Monitoreo y Sistema de Detección de Intrusiones

Tomás F. Melli

September 2025

Índice

1 Network Security Monitoring (NSM)

Network Security Monitoring (NSM) es un enfoque de seguridad informática que se centra en la recolección, análisis y escalamiento de información relacionada con eventos y actividades que ocurren en una red. A diferencia de soluciones que buscan únicamente bloquear ataques, el NSM se basa en la premisa de que ningún sistema es completamente invulnerable, por lo que resulta imprescindible contar con visibilidad continua del tráfico de red y de las actividades de los sistemas para detectar, comprender y responder a incidentes de seguridad.

El concepto fue popularizado por Richard Bejtlich, quien lo definió como la práctica de la **detección de intrusiones basada en la captura y análisis de datos de red**. NSM se concibe como una estrategia integral que combina registros (logs), tráfico (full packet capture, flujos) y alertas (IDS/IPS), con el fin de construir una visión detallada de lo que sucede en el entorno de red. Además, permite la correlación de eventos a lo largo del tiempo, facilitando la identificación de patrones de ataque avanzados y persistentes (APT).

1.1 Componentes de NSM

Los componentes típicos de un sistema de NSM incluyen:

- **Captura de tráfico de red**: obtención de copias de paquetes o flujos para su posterior inspección.
- **Sistemas de detección de intrusiones (IDS/IPS)**: generan alertas en función de patrones o anomalías. Los IDS pueden ser basados en firmas, heurísticos o análisis de comportamiento.
- **Recolección de logs y eventos**: incluye registros de servidores, dispositivos de red, aplicaciones críticas y sistemas operativos.
- **Herramientas de análisis y correlación**: software especializado que permite unir la información proveniente de diferentes fuentes y generar inteligencia procesable. Esto incluye dashboards, correlación de eventos y análisis forense en tiempo casi real.

El objetivo principal del NSM es proveer a los analistas de seguridad de datos verificables y contextualizados que faciliten:

1. **Detección temprana de incidentes**, permitiendo identificar comportamientos anómalos o ataques en curso.
2. **Análisis forense de eventos pasados**, reconstruyendo la cadena de eventos para entender cómo ocurrió un ataque.
3. **Respuesta oportuna y fundamentada** ante amenazas reales, mejorando la mitigación y la prevención futura.

1.2 Recolección y almacenamiento de tráfico

La efectividad del NSM depende directamente de la calidad y amplitud de la recolección de tráfico. Se pueden distinguir varios niveles de datos, cada uno con ventajas y limitaciones:

- **Contenido completo de paquetes (full packet capture, Tcpdump)** Captura todos los paquetes, incluyendo las capas de aplicación.
  - Permite la máxima granularidad para análisis detallados y forenses.
  - Los paquetes pueden ser procesados posteriormente por diversas herramientas de análisis.
  - Alto costo de almacenamiento y procesamiento.
  - El cifrado protege los datos, pero los encabezados de red y transporte permanecen visibles, permitiendo análisis de metadatos y flujos.
- **Datos de sesiones (Argus, NetFlow o equivalentes)** Resumen de las conversaciones entre sistemas, representando conexiones de red como sesiones en lugar de paquetes individuales.
  - Reduce significativamente la cantidad de datos a almacenar.
  - Independiente de los datos de aplicación, útil incluso cuando el tráfico está cifrado.
  - Facilita el análisis de patrones de comunicación y detección de anomalías en el comportamiento de la red.
  - Interpreta encabezados IP, TCP, UDP e ICMP para generar tablas de sesión que pueden ser procesadas en tiempo real.
- **Datos estadísticos (Capinfos, Tcpdstat)** Proporcionan una visión resumida de eventos, útil para monitoreo de alto nivel y generación de métricas.

  - Permiten detectar tendencias y picos de tráfico que podrían indicar ataques como DDoS.
  - Se usan para auditorías y reportes de rendimiento de la red.
- **Datos de alerta (Snort, otros IDSs)** Alertas generadas por sistemas de detección de intrusiones, ya sea por reglas predefinidas o detección de anomalías.
  - Permiten identificar rápidamente comportamientos sospechosos sin necesidad de revisar todo el tráfico.
  - Complementan la captura de paquetes y datos de sesión, proporcionando contexto inmediato para la respuesta a incidentes.

Propósitos de la recolección de datos

Cada nivel de datos cumple objetivos específicos dentro de NSM:

- **Paquetes completos**: máxima flexibilidad para análisis post-incidente, reconstrucción de sesiones y examen forense.
- **Datos de sesión**: eficiencia en almacenamiento y resiliencia frente a tráfico cifrado, útil para patrones de comunicación y correlación de eventos.
- **Datos estadísticos**: monitoreo de tendencias y eventos a nivel global, detección de anomalías volumétricas.
- **Alertas**: detección inmediata de comportamientos sospechosos, soporte para priorización de incidentes.

Este enfoque jerárquico permite a los equipos de seguridad balancear **precisión de análisis, costos de almacenamiento y velocidad de detección**, lo cual es fundamental en entornos de red modernos con alto volumen de tráfico y amenazas sofisticadas.

1.2.1 Ejemplo de captura con Argus (Datos de Sesiones)

Argus es una herramienta de monitorización de flujo de red que resume las conexiones entre sistemas en formato de sesiones. A continuación se muestra un ejemplo de salida típica de Argus:

- **timestamp**: fecha y hora del evento de red registrado.
- **protocol**: protocolo utilizado (por ejemplo, TCP, UDP o ICMP).
- **src IP**: dirección IP de origen, a veces acompañada del puerto de origen.
- **direction**: dirección de la comunicación, puede indicar flujo bidireccional ($<->$) o unidireccional ($->$).
- **dst IP**: dirección IP de destino, a veces acompañada del puerto de destino.
- **status**: estado de la conexión o tipo de evento asociado (por ejemplo, ECO, FIN, RST, TIM).

```
timestamp      protocol    src IP      direction  dst IP      status

17 Apr 02 09:59:16 icmp 192.172.1.26       <-> 192.172.1.253      ECO
17 Apr 02 09:59:16  tcp 192.172.191.46.458  -> 207.68.162.24.80   FIN
17 Apr 02 09:59:16 icmp 192.172.1.25       <-> 192.172.1.253      ECO
17 Apr 02 09:59:16  tcp 192.18.221.25.119   -> 192.172.191.61.25  FIN
17 Apr 02 09:59:16  tcp 192.172.1.6.3562    -> 209.10.33.195.80   FIN
17 Apr 02 09:59:16  tcp 192.172.1.23.5936   -> 61.200.81.153.80   EST
17 Apr 02 09:59:16  tcp 192.172.191.46.4585 -> 64.4.30.24.80      FIN
17 Apr 02 09:59:17  tcp 192.172.191.46.4990 -> 12.12.162.203.80   RST
17 Apr 02 10:00:04  tcp 192.172.191.46.240  -> 216.33.240.24.80   RST
17 Apr 02 09:59:17  tcp 142.177.221.77.177  -> 192.172.18.27.634  RST
17 Apr 02 10:00:02 icmp 192.172.1.25        -> 192.172.1.253      ECO
17 Apr 02 10:00:02 icmp 129.82.45.220       -> 192.172.1.3        ECO
17 Apr 02 10:00:02 icmp 129.82.45.220       -> 192.172.1.3        ECO
17 Apr 02 10:00:02  udp 205.158.62.41.967   -> 192.172.191.6.53   TIM
17 Apr 02 10:00:02 icmp 129.82.45.220       -> 192.172.1.3        ECO
```

Adicionalmente, Argus puede mostrar la cantidad de paquetes y de bytes transferidos, ofreciendo un resumen cuantitativo de la comunicación de red. Los estados comunes que aparecen en la columna de `status` incluyen:

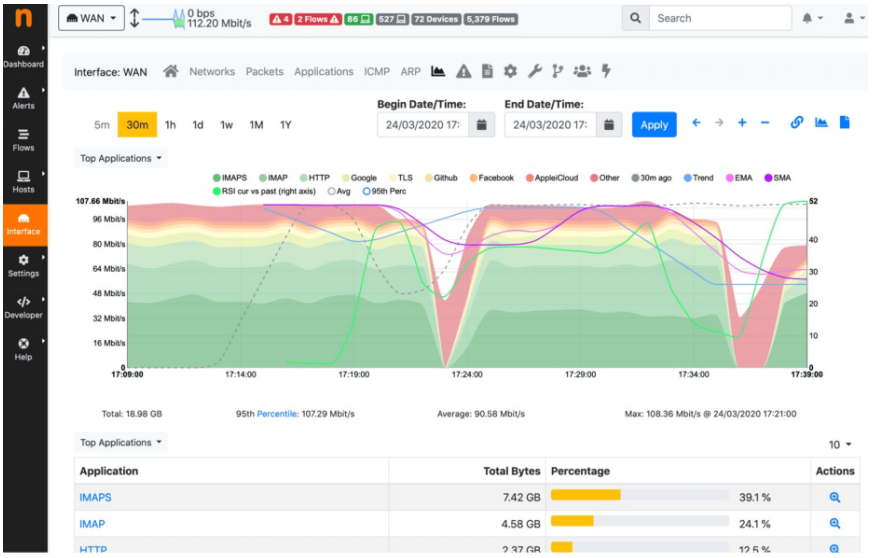
- `ECO`: paquete ICMP de eco (ping).
- `FIN`: finalización de una conexión TCP.
- `RST`: reinicio de una conexión TCP.
- `TIM`: expiración de tiempo o timeout en una sesión.

1.2.2 ntopng (Datos de Sesiones)

ntopng es una aplicación de monitoreo de tráfico de red basada en web. Es la evolución moderna del original *ntop* (Network Top), creado en 1998. ntopng permite a los administradores de sistemas y redes analizar el comportamiento de la red en tiempo real y de manera histórica, ofreciendo visibilidad completa del tráfico.

Características clave

- **Análisis en tiempo real:** Monitorea flujos activos, hosts y protocolos de capa 7 (HTTP, DNS, VoIP, entre otros).
- **Detección de protocolos:** Utiliza nDPI para identificar más de 250 protocolos de aplicación.
- **Monitoreo de rendimiento:** Mide métricas como RTT, latencia, jitter, pérdida de paquetes y MOS en comunicaciones multimedia.
- **Geolocalización de hosts:** Visualiza la ubicación geográfica de los dispositivos conectados.
- **Soporte para múltiples interfaces:** Captura tráfico desde puertos espejo (SPAN), TAPs o mediante NetFlow, sFlow o IPFIX.
- **Alertas y monitoreo activo:** Realiza verificaciones periódicas de conectividad y servicios (ICMP, HTTP, HTTPS, Speedtest).
- **Interfaz web intuitiva:** Accesible desde cualquier navegador, con soporte para múltiples idiomas y temas.



1.2.3 Ejemplos de herramientas (Datos Estadísticos)

Algunas herramientas ampliamente utilizadas para generar estadísticas de tráfico incluyen:

- **tcpdstat:** Analiza archivos `tcpdump` o capturas de paquetes para generar estadísticas de protocolos y hosts.

```
StartTime: Wed Oct  1 18:58:04 2003
EndTime:   Wed Oct  1 20:01:08 2003
# of packets: 7768 (3.33MB)


### IP address Information ###
# of IPv4 addresses: 9
### Protocol Breakdown ###
<<<<
     protocol                packets  bytes            bytes/pkt
-------------------------------------------------------------------------
[0] total              7768 (100.00%)          3496942 (100.00%)    450.17
[1] ip                 7752 ( 99.79%)          3495982 ( 99.97%)    450.98
[2] tcp                7723 ( 99.42%)          3491796 ( 99.85%)    452.13
[3] http(s)             913 ( 11.75%)           816137 ( 23.34%)    893.91
[3] http(c)             302 (  3.89%)            28309 (  0.81%)     93.74
[3] ftp                  11 (  0.14%)              828 (  0.02%)     75.27
[3] other              6497 ( 83.64%)          2646522 ( 75.68%)    407.35
[2] udp                  28 (  0.36%)             4116 (  0.12%)    147.00
[3] other                28 (  0.36%)             4116 (  0.12%)    147.00
[2] icmp                  1 (  0.01%)               70 (  0.00%)     70.00
```

  - **StartTime / EndTime:** Indican el período de tiempo de la captura de tráfico. En el ejemplo: 18:58:04 a 20:01:08 del 1 de octubre de 2003.
  - **Número de paquetes (# of packets):** Total de paquetes capturados y volumen en bytes. Ejemplo: 7768 paquetes, 3.33 MB de datos.
  - **Información de direcciones IP:** Cantidad de direcciones IPv4 únicas involucradas en la captura. Ejemplo: 9 direcciones IPv4.
  - **Desglose por protocolos (Protocol Breakdown):** Muestra estadísticas de cada protocolo detectado:
    * **total:** Todos los paquetes, totalizando 7768 paquetes y 3,496,942 bytes.
    * **ip:** Paquetes IP (7752), casi el total del tráfico capturado.
    * **tcp:** Paquetes TCP (7723), representando 99.42% de los paquetes IP.
    * **http(s):** Paquetes HTTP seguros (913), con 816,137 bytes, promedio 893.91 bytes/paquete.
    * **http(c):** Paquetes HTTP no seguros (302), con 28,309 bytes, promedio 93.74 bytes/paquete.
    * **ftp:** Paquetes FTP (11), con 828 bytes, promedio 75.27 bytes/paquete.
    * **other (TCP):** Otros paquetes TCP (6497), 2,646,522 bytes, promedio 407.35 bytes/paquete.
    * **udp:** Paquetes UDP (28), 4116 bytes, promedio 147 bytes/paquete.
    * **other (UDP):** Todos los paquetes UDP no clasificados, coinciden con los anteriores (28 paquetes).
    * **icmp:** Paquetes ICMP (1), 70 bytes, promedio 70 bytes/paquete.

- **Wireshark:** Además de análisis detallado de paquetes, permite generar estadísticas de protocolos, conversaciones y flujos de red.
- **ntop/ntopng:** Ofrece estadísticas en tiempo real sobre tráfico de red, uso de protocolos y volumen de datos por host o por aplicación.

2 IDS – Sistemas de Detección de Intrusiones

La **Detección de Intrusiones** se refiere al proceso de monitorear los eventos que ocurren en un sistema o red de computadoras, analizándolos en busca de señales que indiquen incidentes de seguridad. Los *Intrusion Detection Systems* (IDS) se pueden comparar con sistemas de seguridad en otras áreas, como alarmas anti-robo o sistemas de vigilancia con videocámaras, cuya función principal es identificar situaciones que podrían constituir una amenaza.

2.1 Problemas comunes de los IDS

A pesar de su utilidad, los IDS enfrentan varios desafíos que deben ser considerados al implementarlos:

- **Falsos positivos:** Ocurren cuando la herramienta clasifica una acción como una posible intrusión cuando, en realidad, se trata de un comportamiento legítimo. La precisión puede mejorarse ajustando las descripciones de patrones de detección de vulnerabilidades.
- **Falsos negativos:** Se producen cuando un ataque ocurre, pero el IDS no lo detecta. Esto puede deberse a ataques desconocidos o mal configurados.
- **Falsas alarmas o ruido:** Alertas generadas en base a datos que forman parte de un ataque, pero que en realidad no representan peligro. Algunas causas típicas incluyen:
  - El ataque afecta a una plataforma distinta a la atacada.
  - La vulnerabilidad no existe en el objetivo atacado.
  - El ataque no logró alcanzar el objetivo.

2.2 Objetivos de un IDS

Los sistemas de detección de intrusiones buscan cumplir con varios objetivos fundamentales:

- **Detectar una amplia variedad de intrusiones:** Tanto ataques conocidos como desconocidos. Esto implica la necesidad de que el IDS pueda aprender y adaptarse a nuevos ataques o cambios en el comportamiento de los usuarios y sistemas.

- **Detectar intrusiones en un tiempo razonable:** En muchos casos, es crítico que la detección ocurra en tiempo real, especialmente si el sistema responde automáticamente ante una intrusión. Sin embargo, en otros escenarios puede ser suficiente reportar intrusiones ocurridas hace algunos minutos u horas, balanceando la precisión con el impacto en el desempeño del sistema.
- **Presentar la información de manera comprensible:** La información generada por un IDS debe ser clara y accesible para los analistas. Idealmente, se puede usar un indicador binario de intrusión, aunque la información más detallada permite examinar el contexto y las características del supuesto ataque. La interfaz de usuario es crítica cuando se monitorean múltiples sistemas simultáneamente.
- **Ser preciso:** Minimizar tanto falsos positivos como falsos negativos, reduciendo el tiempo requerido para identificar, verificar y responder a ataques. La eficiencia y exactitud del sistema son esenciales para mantener la seguridad sin afectar la operación normal de la red o sistemas monitoreados.

2.3 Clasificación

- **Según el ámbito de ejecución:**
  - **IDS de Host (HIDS):** Monitorean eventos y actividades de un solo equipo.
  - **IDS de Red (NIDS):** Analizan el tráfico que circula por la red completa.
- **Según el método de detección:**
  - **Basado en patrones conocidos:** Identifica ataques comparando el tráfico con firmas previamente registradas.
  - **Basado en heurísticas o estadística:** Detecta comportamientos anómalos que se desvían del patrón normal de operación.

2.4 NIDS - Network-based Intrusion Detection Systems

Los **NIDS (Network-based Intrusion Detection Systems)** son sistemas de detección de intrusiones que monitorean el **tráfico de red** en tiempo real con el objetivo de identificar actividades sospechosas o maliciosas.

A diferencia de los **HIDS** (Host-based IDS), que analizan eventos dentro de un solo equipo, los NIDS supervisan los **paquetes que circulan por segmentos de red**, permitiendo detectar ataques que afectan a múltiples hosts, movimientos laterales y patrones de tráfico inusuales.

En esencia, un NIDS funciona como un "sensor de red", inspeccionando paquetes y flujos de comunicación para alertar sobre posibles violaciones de seguridad, anomalías o intentos de intrusión.

Subclasificación: heurística o estadística

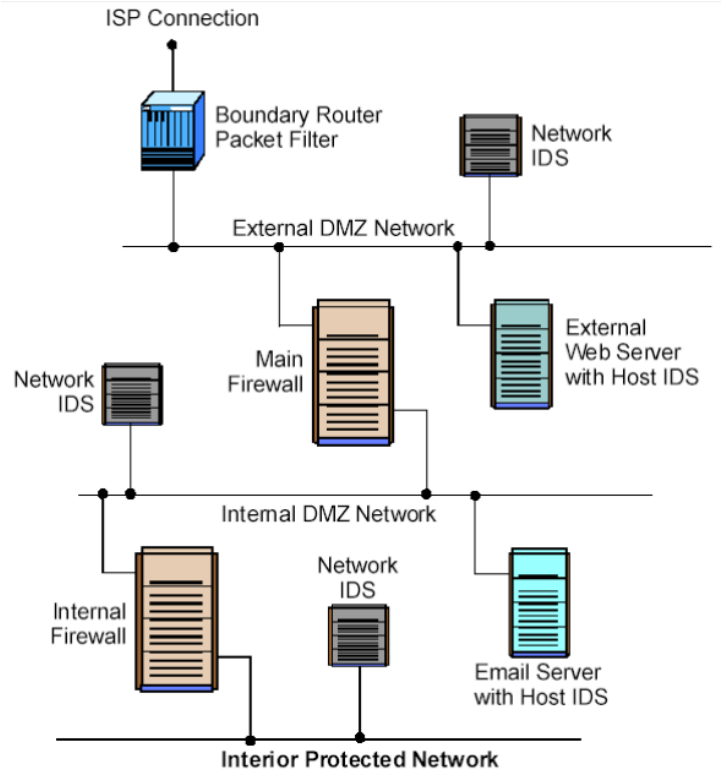
Los NIDS se subclasifican en función de cómo identifican un ataque:

- **Basados en patrones o firmas conocidas:**
  - Buscan coincidencias entre el tráfico de red y **firmas de ataques previamente registradas**.
  - Efectivos contra ataques documentados, pero pueden fallar ante amenazas nuevas o desconocidas.
- **Basados en heurísticas o estadísticas (detección de anomalías):**
  - Analizan el tráfico en busca de **desviaciones respecto a un comportamiento normal**.
  - Utilizan métricas estadísticas, como volumen de paquetes, frecuencia de conexiones o patrones de puerto, para determinar si un evento es sospechoso.
  - Pueden detectar ataques desconocidos, como DoS, escaneos de puertos o intentos de *brute-force*, aunque pueden generar más falsos positivos si los patrones normales no están bien definidos.

2.4.1 Ubicación de un NIDS

La ubicación de un NIDS en la red es crítica para su eficacia. Se recomienda:

- Ubicarlo en segmentos estratégicos donde el tráfico relevante pueda ser monitoreado, como puntos de entrada o salida de la red.
- Puede colocarse detrás de un firewall para inspeccionar tráfico permitido o frente a segmentos internos para detectar movimientos laterales.



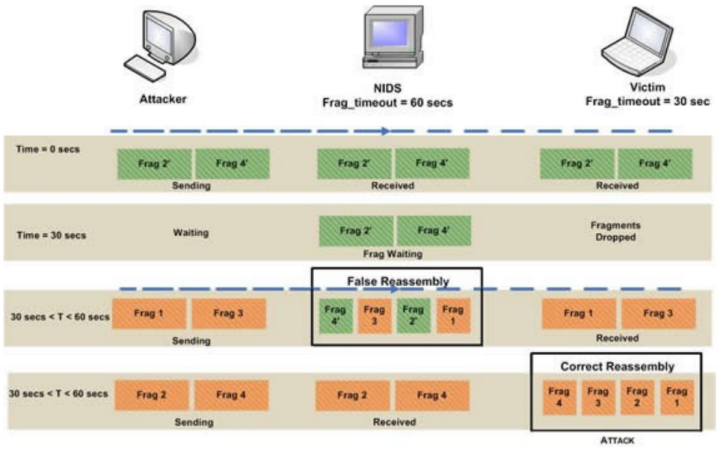
2.4.2 Ataques a NIDS

Incluso los NIDS pueden ser objetivo de técnicas de evasión por parte de atacantes:

- **Inserción:** El NIDS acepta un paquete que el sistema final rechaza, generando alertas que no son relevantes para el objetivo real.
- **Evasión:** El sistema final acepta un paquete que el NIDS descarta por error, permitiendo que ataques pasen desapercibidos.

2.4.3 Ejemplo: Fragmentación IP

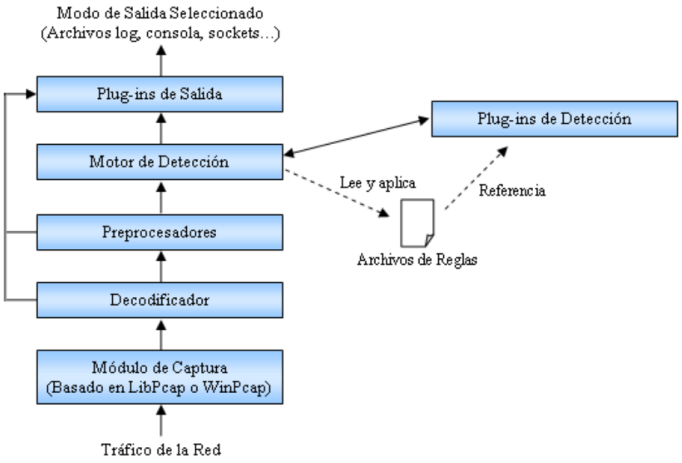
Una técnica común de evasión consiste en la fragmentación de paquetes IP. El atacante divide un paquete en múltiples fragmentos que el NIDS puede no reconstruir correctamente, mientras que el sistema final los ensambla y ejecuta, permitiendo que el ataque se lleve a cabo sin ser detectado.



2.4.4 Ejemplo NIDS: Snort

Snort es un **NIDS basado en patrones** que utiliza reglas para analizar paquetes de red y detectar posibles ataques. Cada regla define un conjunto de condiciones que un evento (paquete, conjunto de paquetes reensamblados o flujo de datos) debe cumplir para generar una alerta.

Arquitectura de Snort



Los componentes básicos de Snort son:

- **Sniffer** o **Módulo de captura**: Captura todos los paquetes que circulan por la red, utilizando la librería `libpcap`.
- **Decodificador:** Construye estructuras de datos a partir de los paquetes capturados, identificando los protocolos de enlace, red, transporte, etc., y detectando posibles problemas en los encabezados.
- **Preprocesadores:** Extienden las funcionalidades del IDS preparando los datos para el motor de detección. Por ejemplo:
  - Detección de escaneos de red.
  - Reensamblado de datos contenidos en múltiples paquetes de una misma sesión.
  - Reensamblado de paquetes IP fragmentados.
  - **HTTP_inspect:** Normaliza las URIs contenidas en solicitudes HTTP, evitando técnicas de evasión que modifican los URIs para confundir al IDS.
- **Motor de detección:** Analiza los paquetes en base a las reglas definidas para identificar ataques conocidos.
- **Postprocesadores o Salida:** Gestionan cómo y dónde se almacenan las alertas y los paquetes relacionados. Pueden generar registros en archivos de texto, enviar traps SNMP, guardar en bases de datos o syslog.

Ejemplos de reglas Snort

- **Regla de ataque a web CGI:**

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-CGI HyperSeehsx.cgi directory traversal attempt";
uricontent:"/hsx.cgi";content:"../../";content:"%00";
flow:to_server,established; reference:bugtraq,2314;
reference:cv,CAN-2001-0253; classtype:web-application-attack;
sid:803; rev:6;)
```

  - `alert tcp`: Indica que la regla genera una alerta para tráfico TCP.
  - `$EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS`: Define el origen y destino de la comunicación.
  - `msg: "..."`: Mensaje descriptivo de la alerta.
  - `uricontent:"/hsx.cgi"`: Busca solicitudes HTTP hacia el script hsx.cgi.
  - `content:"../../"` y `content:"%00"`: Detecta intentos de *directory traversal* y terminadores nulos.
  - `flow:to_server,established`: Evalúa solo flujos hacia el servidor que ya estén establecidos.
  - `reference`: Proporciona referencias externas a vulnerabilidades documentadas.
  - `classtype`: Clasifica la alerta según tipo de ataque.
  - `sid` y `rev`: Identificador único de la regla y número de revisión.

- **Regla P2P BitTorrent:**

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any
(msg:"P2P BitTorrent announce request";
flow:to_server,established; content:"GET"; depth:4;
content:"/announce"; distance:1; content:"info_hash="; offset:4;
content:"event=started"; offset:4;
classtype:policy-violation; sid:2180; rev:2;)
```

  - Detecta solicitudes de *announce* de BitTorrent, típicas de clientes P2P.
  - La regla analiza solo flujos establecidos hacia servidores externos.
  - Los parámetros `depth`, `distance` y `offset` ajustan la búsqueda de cadenas dentro del paquete, permitiendo identificar con precisión patrones en la carga útil.
  - Clasificación como violación de política, útil para redes corporativas que restringen tráfico P2P.

- **Regla MS15-034:**

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80
(msg:"MS15-034 Range Header HTTP.sys Exploit";
content:"|0d 0a| Range: bytes="; nocase;
content:"-"; within:20; byte_test:10,>,1000000000,0,relative,string,dec;
sid:1001239;)
```

  - Detecta intentos de explotación de la vulnerabilidad HTTP.sys MS15-034 en servidores Windows.
  - `content:"|0d 0a| Range: bytes="`: Busca la cabecera específica del ataque.
  - `nocase`: Ignora mayúsculas/minúsculas.
  - `within:20`: Limita la búsqueda a los primeros 20 bytes tras la coincidencia anterior.
  - `byte_test`: Evalúa valores numéricos en la carga útil para identificar rangos anómalos que indican explotación.
  - `sid`: Identificador único de la regla.

Recursos adicionales

- Reglas open source: https://rules.emergingthreats.net/open/
- Preprocesadores y técnicas de evasión: https://blog.didierstevens.com/2015/04/17/ms15-034-detection-some-observa

Alertas generadas por Snort

Cuando una regla coincide con el tráfico, Snort genera una alerta. Por ejemplo:

```
[**] [1:1444:3] TFTP Get [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
10/30-12:11:07.031155 200.59.73.153:33206 -> 200.59.103.1:69
UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:50 DF
Len: 22
```

- `[**] [1:1444:3] TFTP Get [**]`: Identificador de alerta y nombre de la regla.
- `Classification: Potentially Bad Traffic`: Clasificación del tipo de actividad.
- `Priority: 2`: Nivel de prioridad de la alerta.
- `10/30-12:11:07.031155`: Fecha y hora del evento.
- `200.59.73.153:33206 -> 200.59.103.1:69`: IP de origen y destino, puertos y protocolo.
- `UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:50 DF Len:22`: Información detallada del paquete, incluyendo longitud y TTL.

Otro ejemplo de alerta:

```
[**] [1:2404:6] NETBIOS SMB-DS Session Setup AndX request
unicode username overflow attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
10/30-12:15:18.219433 200.59.105.60:1431 -> 200.59.73.153:445
TCP TTL:46 TOS:0x0 ID:42991 IpLen:20 DgmLen:1484 DF
***AP*** Seq:0xE03BC43C Ack:0xBFA96C1B Win:0xFFBF TcpLen:20
[Xref => http://www.eeye.com/html/Research/Advisories/AD20040226.html]
[Xref => http://www.securityfocus.com/bid/9752]
```

- Indica un intento de desbordamiento de buffer en NetBIOS SMB.
- Muestra información TCP detallada, incluyendo número de secuencia (`Seq`), acuse de recibo (`Ack`) y tamaño de ventana (`Win`).

2.4.5 Zeek – Plataforma de Análisis de Tráfico de Red

Zeek es una plataforma integral para el análisis del tráfico de red. Aunque a menudo se lo compara con sistemas clásicos de detección o prevención de intrusiones (NIDS/NIPS), Zeek adopta un enfoque diferente: en lugar de limitarse a alertar sobre ataques conocidos, proporciona un marco flexible que permite a los usuarios realizar un **monitoreo personalizado y en profundidad** mucho más allá de las capacidades de los sistemas tradicionales.

Zeek genera logs detallados de las **transacciones** de varios protocolos, tales como:

- HTTP
- SSH
- SMTP
- DNS
- Certificados SSL/TLS

Estos logs permiten un análisis forense y de comportamiento de red altamente granular, útil tanto para la seguridad como para la administración de la infraestructura.

Ejemplo de log Zeek

A continuación se muestra un ejemplo de un log HTTP generado por Zeek:

```
2013-01-16T19:09:47+0000 90E6goBBSw3 192.168.238.152 41482
217.160.51.31 80 1 GET www.testmyids.com / -
Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:18.0) Gecko/20100101 Firefox/18.0 0 39 200 OK - - - (empty) - - - t

2013-01-16T19:09:47+0000 90E6goBBSw3 192.168.238.152 41482
217.160.51.31 80 2 GET www.testmyids.com /favicon.ico -
Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:18.0) Gecko/20100101 Firefox/18.0 0 640 404 Not Found - - (empty) -
```

Desglose del log

Cada campo del log tiene un significado específico:

- **Timestamp (2013-01-16T19:09:47+0000)**: Fecha y hora en formato UTC cuando se registró la transacción.
- **ID de conexión (90E6goBBSw3)**: Identificador único de la sesión de red que permite correlacionar varios eventos del mismo flujo.
- **IP de origen y puerto (192.168.238.152 41482)**: Dirección IP y puerto del cliente que inició la conexión.
- **IP de destino y puerto (217.160.51.31 80)**: Dirección IP y puerto del servidor web al que se realizó la petición.
- **Número de solicitud en la conexión (1, 2)**: Permite identificar múltiples solicitudes dentro de la misma sesión.
- **Método HTTP (GET)**: Tipo de solicitud realizada al servidor.
- **Host y URI (www.testmyids.com /, /favicon.ico)**: Servidor y recurso solicitado.
- **User-Agent**: Información del cliente que realiza la petición (navegador, sistema operativo, versión).

- **Flags o códigos adicionales (0 39, 0 640)**: Tamaño de la solicitud y respuesta, útil para analizar tráfico y detectar anomalías.
- **Código HTTP y mensaje (200 OK, 404 Not Found)**: Resultado de la solicitud, indicando éxito o error.
- **Tipo de contenido (text/plain, text/html)**: Muestra el tipo MIME de la respuesta.

Análisis del ejemplo

- La primera línea indica una solicitud GET exitosa a la página raíz del sitio `www.testmyids.com`, con respuesta HTTP 200 OK y contenido tipo `text/plain`.
- La segunda línea corresponde a la solicitud de un recurso inexistente (`/favicon.ico`), generando un error HTTP 404 Not Found, con tipo de contenido `text/html`.
- Ambos eventos pertenecen a la misma conexión, como lo indica el ID de sesión, lo que permite a Zeek reconstruir el flujo completo de la transacción.
- Este nivel de detalle facilita la detección de patrones sospechosos, análisis forense y auditoría del comportamiento de usuarios y aplicaciones.

2.5 HIDS

2.5.1 Swatch

Swatch (Simple WATCHer) es un HIDS orientado al monitoreo de logs en sistemas UNIX. Originalmente diseñado como un "watchdog" para revisar la actividad registrada por `syslog`, Swatch ha evolucionado para permitir la supervisión de otros tipos de logs, ofreciendo funcionalidades como:

- Definir patrones de entradas a **resaltar** o **ignorar**.
- Notificaciones en tiempo real mediante correo electrónico o ejecución de comandos.
- Monitoreo continuo y configurable para adaptarse a distintos tipos de eventos de seguridad.

2.5.2 Logcheck

Logcheck es una herramienta similar a Swatch, optimizada para sistemas Debian. Su funcionamiento principal consiste en:

- Ejecutarse periódicamente (normalmente cada hora).
- Analizar los logs del sistema en busca de entradas anormales o indicativas de problemas de seguridad.
- Generar un reporte con los eventos detectados, que puede ser enviado automáticamente por correo electrónico.
- Utilizar reglas actualizadas para filtrar y clasificar eventos de seguridad.

Ejemplo de log con Logcheck

```
Security Events
Aug 1 09:09:34 matute sshd[1401]: (pam_unix) authentication failure;
logname= uid=0 euid=0 tty=ssh ruser= rhost=10.xx.yy.26
Aug 1 09:09:36 matute sshd[1401]: error: PAM: User not known
to the underlying authentication module for illegal user n3ssus from 10.xx.yy.26

System Events
Aug 1 09:21:31 matute sshd[2473]: Did not receive identification string from 10.xx.yy.26
Aug 1 09:21:37 matute sshd[2474]: Bad protocol version identification 'GET / HTTP/1.0' from 10.xx.yy.26
Jul 30 11:55:43 matute Inscripcion[25721]: desde 200.xx.yy.132 - invalid: cdi:20-12430832|
```

Desglose del log

Cada entrada del log contiene información relevante para el análisis de seguridad:

- **Timestamp**: Fecha y hora del evento, por ejemplo `Aug 1 09:09:34`.
- **Hostname**: Nombre del equipo donde ocurrió el evento, en este caso `matute`.
- **Servicio y PID**: Indica qué servicio generó la entrada y el identificador de proceso, por ejemplo `sshd[1401]`.
- **Mensaje de evento**:
  - `(pam_unix) authentication failure`: Fallo de autenticación en SSH mediante PAM.
  - `User not known to the underlying authentication module`: Intento de login con usuario inexistente.
  - `Did not receive identification string`: Conexión SSH incompleta, posiblemente escaneo o ataque automatizado.
  - `Bad protocol version identification`: Uso incorrecto de protocolo en intento de conexión HTTP/SSH.
  - `invalid cdi`: Evento de sistema indicando datos inválidos provenientes de una IP externa.
- **IP origen**: Dirección desde la cual se realizó la acción, útil para identificar posibles atacantes.

Análisis del ejemplo

- Los primeros eventos indican intentos de acceso no autorizado mediante SSH, fallos de autenticación y usuarios inexistentes.
- Los eventos posteriores muestran conexiones incompletas o con protocolos incorrectos, típicos de escaneos de puertos o pruebas automatizadas.
- Logcheck permite agrupar y filtrar estos eventos, facilitando que el administrador reciba un reporte consolidado y pueda tomar medidas preventivas.
- Este tipo de HIDS basado en logs es especialmente útil para monitorear sistemas críticos y detectar comportamientos anómalos sin necesidad de inspeccionar el tráfico de red completo.

2.5.3 Tripwire y AIDE

Tripwire y **AIDE** (Advanced Intrusion Detection Environment) son HIDS que monitorean la integridad de los archivos en un sistema. Permiten detectar cambios, adiciones o eliminaciones de archivos, alertando al administrador sobre posibles incidentes de seguridad.

Funcionamiento

- Estas herramientas generan una base de datos con información de cada archivo, incluyendo:
  - Tamaño del archivo.
  - Número de bloques ocupados.
  - Fecha y hora de modificación y cambio de metadatos.
  - Inode del archivo.
  - Hashes criptográficos (MD5, SHA1).
- Periódicamente comparan el estado actual con la base de datos previa para identificar:
  - Archivos nuevos.
  - Archivos eliminados.
  - Cambios en atributos o contenido.
- Generan alertas automáticas detalladas para que el administrador pueda revisar y actuar.

Ejemplo de reporte AIDE

```
Envelope-to: root@localhost.localdomain
Delivery-date: Fri, 14 Apr 2006 06:25:16 -0300
To: root@localhost.localdomain
Subject: Daily AIDE report for localhost.localdomain
From: root <root@localhost.localdomain>
This is an automated report generated by the Advanced Intrusion Detection
Environment on localhost.localdomain at 2006-04-14 06:25.

added:/usr/bin/tftp
File: /bin/tar
Size : 163820, 163852
Bcount: 328, 336
Mtime : 2004-08-03 11:31:59, 2006-02-24 18:21:24
Ctime : 2006-01-27 09:24:39, 2006-03-27 19:56:39
Inode : 2326559, 2326584
MD5   : lqHdZO5kJKbPp4OQFdmNZw==, 0O0qYuZFk3VRPSEaK8Xp+w==
SHA1  : sB7B1di8CecXlpNZ16kw/kSLVVs=, Xu6u1r5mPWNOT4iB3w4Dp9KRp58=
```

Desglose del reporte

Cada campo del reporte indica información crítica sobre los archivos monitoreados:

- **added:/usr/bin/tftp**: Se detectó un archivo agregado recientemente en el sistema.
- **File: /bin/tar**: Archivo examinado por AIDE.
- **Size**: Tamaño en bytes de cada archivo (`163820` y `163852`).
- **Bcount**: Número de bloques ocupados por cada archivo (`328` y `336`).
- **Mtime**: Fecha y hora de la última modificación del contenido de los archivos (`2004-08-03 11:31:59` y `2006-02-24 18:21:24`).
- **Ctime**: Fecha y hora del último cambio en metadatos (`2006-01-27 09:24:39` y `2006-03-27 19:56:39`).
- **Inode**: Número de inode, útil para identificar archivos de manera única (`2326559` y `2326584`).
- **MD5 / SHA1**: Hashes criptográficos de los archivos, usados para verificar integridad (`lqHdZO5kJKbPp4OQFdmNZw== / 0O0qYuZFk3VRPSEaK8Xp+w==` y `sB7B1di8CecXlpNZ16kw/kSLVVs= / Xu6u1r5mPWNOT4iB3w4Dp9KRp58=`).

Análisis del ejemplo

- Se detectaron cambios en dos archivos: uno agregado (`/usr/bin/tftp`) y uno monitorizado (`/bin/tar`).
- La comparación con la base de datos permite identificar modificaciones, adiciones o posibles manipulaciones.
- Los hashes aseguran que incluso cambios mínimos en el contenido serán detectados.
- Este tipo de monitoreo es crítico en sistemas sensibles para prevenir accesos o modificaciones no autorizadas.

2.5.4 OSSEC

OSSEC es un HIDS open source que combina varias funcionalidades de seguridad: análisis de logs, chequeo de integridad de archivos, monitoreo del registro de Windows, detección de rootkits, y alertas en tiempo real. Corre en Linux, OpenBSD, FreeBSD, MacOS, Solaris y Windows.

Funcionamiento

- OSSEC monitoriza logs de aplicaciones y del sistema operativo para detectar actividad sospechosa.
- Evalúa reglas definidas y genera alertas según el nivel de severidad de cada evento.
- Permite centralizar el monitoreo de múltiples hosts y generar reportes automáticos.

Ejemplos de notificaciones OSSEC

```
OSSEC HIDS Notification. 2007 Aug 14 13:20:23
Received From: monitor->/var/log/apache2/error.log
Rule: 30109 fired (level 9) -> "Attempt to login using a non-existent user."
Portion of the log(s):
[Tue Aug 14 13:20:23 2007] [error] [client 10.xx.yy.51] user test not found:
/nagios2/, referer: http://monitor.arcert.gov.ar/

OSSEC HIDS Notification. 2007 Aug 14 13:08:05
Received From: monitor->/var/log/auth.log
Rule: 5701 fired (level 12) -> "Possible attack on the ssh server (or version gathering)."
Portion of the log(s):
Aug 14 13:08:04 monitor sshd[21642]: Bad protocol version identification 'quit' from UNKNOWN

OSSEC HIDS Notification. 2006 Sep 06 23:15:21
Received From: (xx) 1.2.3.4->/usr/pages/xx/logs/web.access_log
Rule: 31151 fired (level 10) -> "Multiple web server 400 error codes from same source ip."
Portion of the log(s):
64.46.38.151 - - [06/Sep/2006:23:14:41 -0300] "POST /xmlsrv/xmlrpc.php HTTP/1.1" 404 223 "-" "Internet Explore
64.46.38.151 - - [06/Sep/2006:23:14:41 -0300] "POST /xmlrpc/xmlrpc.php HTTP/1.1" 404 223 "-" "Internet Explore
64.46.38.151 - - [06/Sep/2006:23:14:40 -0300] "POST /xmlrpc.php HTTP/1.1" 404 216 "-" "Internet Explorer 6.0"
64.46.38.151 - - [06/Sep/2006:23:14:39 -0300] "POST /xmlrpc.php HTTP/1.1" 404 216 "-" "Internet Explorer 6.0"
64.46.38.151 - - [06/Sep/2006:23:13:52 -0300] "POST /xmlsrv/xmlrpc.php HTTP/1.1" 404 223 "-" "Internet Explore
64.46.38.151 - - [06/Sep/2006:23:13:52 -0300] "POST /xmlrpc/xmlrpc.php HTTP/1.1" 404 223 "-" "Internet Explore
64.46.38.151 - - [06/Sep/2006:23:13:50 -0300] "POST /xmlrpc.php HTTP/1.1" 404 216 "-" "Internet Explorer 6.0"

OSSEC HIDS Notification. 2006 Oct 24 18:46:29
Received From: (xx) 200.1.2.a->/var/log/maillog
Rule: 3354 fired (level 12) -> "Multiple misuse of SMTP service (bad sequence of commands)."
Portion of the log(s):
postfix/smtpd[6741]: NOQUEUE: reject: RCPT from unknown[201.82.55.24]:
503 <nplxfbtk@fbi.com>: Sender address rejected: Improper use of SMTP command pipelining; from=<nplxfbtk@fbi.c
postfix/smtpd[6741]: NOQUEUE: reject: RCPT from unknown[201.82.55.24]:
503 <nplxfbtk@fbi.com>: Sender address rejected: Improper use of SMTP command pipelining; from=<nplxfbtk@fbi.c
postfix/smtpd[6741]: NOQUEUE: reject: RCPT from unknown[201.82.55.24]:
503 <nplxfbtk@fbi.com>: Sender address rejected: Improper use of SMTP command pipelining; from=<nplxfbtk@fbi.c
```

Desglose de los logs

- **OSSEC HIDS Notification**: Identifica la alerta generada por una regla disparada.
- **Fecha y hora**: Momento exacto en que se detectó el evento.
- **Received From**: Archivo de log o sistema que generó la información.
- **Rule y Level**: Número de regla y nivel de severidad (mayor número = mayor gravedad).
- **Mensaje de la regla**: Explicación breve del tipo de evento detectado.
- **Portion of the log(s)**: Fragmento del log que disparó la alerta, mostrando:
  - Dirección IP origen.
  - Servicio o demonio afectado (apache2, sshd, postfix, etc.).
  - Tipo de error o actividad sospechosa (login fallido, error 404, SMTP mal usado, etc.).
  - URI, comando, o recurso involucrado.
  - Agente de usuario y otra información contextual (referer, headers, etc.).

Análisis

- OSSEC permite detectar intrusiones, errores o comportamientos anómalos en servidores web, SSH y correo.
- Cada alerta muestra el nivel de severidad y evidencia concreta para que el administrador pueda actuar.

- La combinación de monitoreo de logs, integridad de archivos y análisis de patrones lo hace un HIDS integral y confiable.

2.5.5 Security Onion

Security Onion es una distribución de Linux orientada a profesionales de la ciberseguridad. Ofrece una plataforma unificada que combina monitoreo de redes, detección de intrusiones, análisis de amenazas y gestión de registros. Su objetivo es proporcionar capacidades de clase empresarial sin costo, permitiendo a las organizaciones monitorear y defender sus infraestructuras de manera efectiva.

3 IPS – Sistemas de Prevención de Intrusiones

Los Sistemas de Prevención de Intrusiones (IPS, por sus siglas en inglés) son herramientas de seguridad que no sólo detectan ataques, sino que también pueden detenerlos antes de que lleguen a su objetivo. A diferencia de los IDS, que se limitan a generar alertas, los IPS actúan de manera proactiva para bloquear o mitigar incidentes de seguridad en tiempo real.

3.1 Características generales

- **Detección y prevención:** Los IPS pueden analizar el tráfico de red o las acciones en un host, identificar patrones de ataque conocidos o anomalías, y tomar medidas automáticas para detener la intrusión.
- **Niveles de operación:**
  - **IPS de red:** Se colocan directamente en la infraestructura de red, generalmente en modo *in-line* o *bridge*, de manera que todo el tráfico pasa a través del sistema.
  - **IPS de host:** Monitorean la actividad directamente sobre un equipo específico, analizando logs, cambios en archivos y comportamientos sospechosos.

3.2 IPS de red

Los IPS de red actúan en tiempo real, filtrando o modificando paquetes según las reglas definidas para prevenir ataques. Su colocación *in-line* les permite bloquear tráfico malicioso antes de que alcance su destino. A menudo, utilizan las mismas reglas que un NIDS, pero con capacidades extendidas para tomar acciones correctivas.

- **Ejemplo de implementación: Snort_Inline**
  - Utiliza las reglas de Snort, pero permite definir acciones adicionales como:
    - `block`: impedir que el paquete llegue al destino.
    - `reject`: descartar el paquete y enviar un mensaje de rechazo.
    - `sdrop`: descartar el paquete sin notificación.
    - `replace`: modificar el contenido del paquete antes de enviarlo.

3.3 Ejemplo de regla Snort_Inline

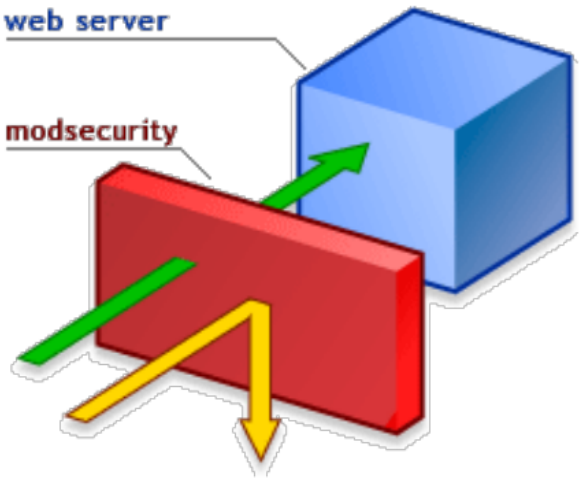
```
drop tcp $EXTERNAL_NET any -> $SMTP_SERVERS 25
(msg:"SMTP AUTH user overflow attempt";
flow:to_server,established;
content:"AUTH"; nocase;
pcre:"/^AUTH\s+\S+\s+[^\n]{128}/mi";
reference:bugtraq,13772;
classtype:attempted-admin; sid:3824; rev:1;)
```

Explicación detallada

- **drop tcp:** indica que el paquete será descartado si cumple las condiciones de la regla.
- **$EXTERNAL_NET any -¿ $SMTP_SERVERS 25:** define el flujo de tráfico que se analizará, desde cualquier host externo hacia los servidores SMTP en el puerto 25.
- **msg:** mensaje que describe el propósito de la regla, en este caso, detectar intentos de *overflow* en la autenticación SMTP.

- **flow:to_server,established:** la regla aplica únicamente a paquetes enviados hacia el servidor en sesiones TCP establecidas.
- **content:"AUTH"; nocase:** busca la cadena `AUTH` sin distinguir mayúsculas o minúsculas.
- **pcre:** expresión regular que detecta patrones específicos de *overflow* en la autenticación.
- **reference:** enlace a la base de vulnerabilidades correspondiente (Bugtraq 13772).
- **classtype:** clasifica la regla como un intento de administración no autorizado.
- **sid y rev:** identificador único de la regla y su versión.

3.4 ModSecurity — WAF para Aplicaciones Web



ModSecurity es un motor de detección y prevención de intrusiones orientado a aplicaciones web (WAF, Web Application Firewall). Originalmente concebido como un módulo para Apache, su funcionalidad evolucionó hasta una librería independiente (*libModSecurity*, ModSecurity v3) con conectores para Apache, Nginx e IIS. El propósito de ModSecurity es aumentar la seguridad de aplicaciones web, protegiéndolas frente a ataques conocidos y emergentes mediante un conjunto de reglas configurables y potentes capacidades de inspección del tráfico HTTP.

3.4.1 Resumen funcional

- Intercepta y analiza peticiones HTTP *antes* de que sean procesadas por la aplicación web.
- Inspecciona cuerpos de petición (POST, multipart/form-data, JSON, etc.) y cuerpos de respuesta cuando está configurado.
- Permite capturar, almacenar y opcionalmente validar archivos subidos por el usuario.
- Ejecuta contramedidas anti-evasión (normalizaciones y transformaciones) y reglas configurables para detectar inyección SQL, XSS, RFI/LFI, CSRF, path traversal, y más.
- Opera en modos `DetectionOnly` (solo alerta), `On` (bloqueo activo) y `Off` (deshabilitado).
- Integra conjuntos de reglas estandarizados (por ejemplo OWASP CRS) y reglas personalizadas.

3.4.2 Modos de despliegue

- **Módulo embebido (Apache mod_security2)**: ejecuta ModSecurity como módulo dentro del servidor web.
- **Librería con conector (ModSecurity v3)**: *libModSecurity* + conector para Nginx/Apache/IIS; separación entre motor y servidor.
- **Reversa/Proxy**: deploy en front-end (reverse proxy) para proteger múltiples backends.
- **Inline con otros componentes**: junto a balanceadores, WAF especializados o proxies inversos.

3.4.3 Componentes y flujo de procesamiento

- **Parser/Decoder**: decodifica encabezados, parámetros y cuerpos (form-data, urlencoded, JSON, multipart).
- **Variables**: interfaz para acceder a elementos del request/response (HEADERS, ARGS, REQUEST_BODY, REQUEST_URI, FILES, RESPONSE_BODY, etc.).
- **Transformaciones**: normalizaciones aplicables a variables (lowercase, urlDecode, htmlEntityDecode, removeNulls, compressWhitespace, etc.) para mitigar evasiones.
- **Motor de reglas (SecRule)**: evalúa condiciones, aplica acciones (log, deny, pass, redirect, proxy-chain, capture, tag, etc.).
- **Logging / Audit**: almacena eventos en `auditlog` (múltiples fases y formatos; puede producir JSON).

3.4.4 Características destacadas

- **Inspección profunda**: análisis de headers, URI, query string, cuerpo de petición y respuesta.
- **Gestión de uploads**: escaneo y hashing de archivos cargados; límite de tamaño y bloqueo de tipos peligrosos.
- **Prevención de evasión**: reglas y transformaciones que detectan encoding, fragmentación y otras técnicas evasivas.
- **Integración con CRS (OWASP Core Rule Set)**: reglas comunitarias para amenazas comunes.
- **Auditoría completa**: registro detallado de requests/ responses cuando es necesario para forense.
- **Soporte para detección basada en anomalías y puntaje**: CRS y reglas pueden aplicar scoring/anomaly handling para decisiones más contextuales.

3.4.5 Configuración y parámetros importantes

- `SecRuleEngine` = {On—DetectionOnly—Off} — modo de operación.
- `SecRequestBodyAccess` / `SecResponseBodyAccess` — habilitan la inspección de cuerpos.
- `SecRequestBodyLimit`, `SecRequestBodyNoFilesLimit` — tamaño máximo de cuerpos a procesar.
- `SecRequestBodyInMemoryLimit` — cuando usar disco vs memoria.
- `SecAuditLog` — ruta y formato del log de auditoría.
- `SecDefaultAction` — acción por defecto aplicada a reglas (por ejemplo `log,phase:2,deny,status:403`).
- `SecRuleRemoveById` / `SecAction` — para deshabilitar o ajustar reglas específicas en tiempo de ejecución.

Reglas: sintaxis básica y explicación

La forma moderna de crear reglas en ModSecurity usa `SecRule` (la sintaxis antigua `SecFilterSelective` era de versiones previas y está obsoleta en v2/v3). Una regla típica:

```
SecRule REQUEST_HEADERS:User-Agent "@rx evilbot" \
    "id:1001,phase:1,deny,log,msg:'Blocked EvilBot User-Agent'"
```

Desglose:

- `SecRule <VARIABLE> <OPERATOR> "<ACTIONS>"`:

- `<VARIABLE>` ejemplo: `REQUEST_METHOD`, `ARGS`, `REQUEST_BODY`, `REQUEST_HEADERS`, `FILES_SIZES`, `ARGS_NAMES`, `XML`, `RESPONSE_BODY`...
- `<OPERATOR>` puede ser `@rx` (regex), `@contains`, `@streq`, `@pm` (pattern match), `@eq`, etc.
- `<ACTIONS>` lista separada por comas: `id:<num>`, `phase:<1|2|3|4|5>`, `log|deny|pass|allow|redirect`, `t:<transform>` (transformación), `msg:'...'`, `tag:'...'`.

Ejemplos

```
# Allow supported request methods only.
SecRule REQUEST_METHOD "!^(GET|HEAD|POST)$" \
    "id:10010,phase:1,log,deny,status:405,msg:'Method not allowed'"

# Require the Host header field to be present.
SecRule REQUEST_HEADERS:Host "^$" \
    "id:10011,phase:1,log,deny,msg:'Missing Host header'"

# SQL injection (básico)
SecRule ARGS "@rx (?i)delete\s+from" \
    "id:10020,phase:2,log,deny,msg:'SQLi attempt - delete from'"

# Command "id" in OUTPUT (response body)
SecRule RESPONSE_BODY "@rx uid=\d+\([A-Za-z0-9]+\)\s+gid=\d+" \
    "id:10030,phase:4,log,deny,msg:'Server disclosing system id output'"

# User agent collector (block known bad bot)
SecRule REQUEST_HEADERS:User-Agent "@pm autoemailspider" \
    "id:10040,phase:1,log,deny,msg:'Blocked known bad UA autoemailspider'"
```

Notas:

- `phase:1` = procesamiento de headers y URI (antes de cuerpo).
- `phase:2` = procesamiento del cuerpo de la petición (POST data).
- `phase:4` = procesamiento de la respuesta (si `SecResponseBodyAccess` está activo).
- Usar `t:lowercase` o `t:urlDecode` en `actions` ayuda a normalizar.

4 Honeypots y Honeytokens

4.1 Concepto y propósito

Un **honeypot** es un recurso informático deliberadamente expuesto o simulado con el objetivo de atraer, detectar y estudiar actividades maliciosas. No está destinado a prestar un servicio legítimo, sino a comportarse como un señuelo que permita:

- Detectar ataques y sondas dirigidas a la red.
- Recolectar inteligencia sobre vectores de ataque, herramientas, payloads y TTPs (tácticas, técnicas y procedimientos).
- Desacoplar el análisis forense del tráfico malicioso del resto de la infraestructura productiva.
- Engañar y retrasar al atacante para ganar tiempo de respuesta.

Un **honeytoken** es un señuelo digital (un dato, credencial, documento o API key falsa) cuyo objetivo es detectar uso no autorizado: cualquier acceso o interacción con el honeytoken se considera indicio de compromiso.

4.2 Clasificación según nivel de interacción

La clasificación principal distingue **honeypots de baja interacción** y **de alta interacción**, cada uno con un perfil de riesgo/costo/información distinto.

Baja interacción

- **Descripción:** Emulan servicios, aplicaciones o respuestas de sistemas operativos pero sin ejecutar una pila real completa. Ofrecen interfaces limitadas que hacen creer al atacante que está interactuando con un servicio real.
- **Ventajas:** Bajo riesgo (el atacante tiene menos posibilidad de pivotar desde el honeypot), fáciles de desplegar y mantener, menor consumo de recursos.
- **Limitaciones:** Capturan información limitada (no permiten analizar a fondo exploits complejos o shells interactivos).
- **Usos típicos:** Detección de escaneos de puertos, vermes, intentos automáticos de explotación y recolección de exploits de red.

Alta interacción

- **Descripción:** Ejecutan servicios, aplicaciones y sistemas operativos reales (máquinas virtuales o físicas). Permiten que el atacante ejecute código, obtenga shells y realice movimientos laterales en un entorno controlado.
- **Ventajas:** Recolección de inteligencia muy rica (comportamiento post-explotación, herramientas del atacante, persistencia, exfiltración).
- **Riesgos y costos:** Alto riesgo (el atacante puede comprometer el honeypot y usarlo como plataforma de ataque si no está correctamente contenida), requieren mucho mantenimiento, monitoreo y esfuerzo para instrumentar y recuperar evidencia forense.
- **Usos típicos:** Estudios avanzados de amenazas, análisis dinámico de malware, interacción con atacantes reales para entender su modus operandi.

4.3 Ejemplos representativos

- **Honeyd** (http://www.honeyd.org) — honeypot de baja interacción que emula máquinas virtuales y servicios, permite simular diferentes stacks y sistemas operativos en la red.
- **Dionaea** (https://github.com/DinoTools/dionaea) — honeypot de baja interacción enfocado en emular vulnerabilidades explotadas por worms para capturar malware y muestras.
- **Cowrie** (https://github.com/cowrie/cowrie) — honeypot de interacción media/alta para SSH y Telnet; captura sesiones, comandos, credenciales y archivos transferidos.
- **Conpot** (https://github.com/mushorg/conpot) — honeypot orientado a sistemas de control industrial (SCADA/ICS).
- **Snare/Tanner** — ejemplos de web-application honeypots (simulan aplicaciones web).
- **T-Pot** (https://github.com/telekom-security/tpotce) — distribución que agrupa múltiples honeypots (Dionaea, Cowrie, Conpot, etc.) + stack de visualización y análisis.
- **Thug** — honeyclient de baja interacción que simula un navegador para visitar URLs maliciosas y analizar JavaScript/drive-by downloads.
- **Honeytokens** — no es una herramienta única; pueden implementarse como cuentas de usuario falsas, claves API, documentos-trampa con links únicos o entradas falsas en bases de datos.

4.4 Cómo funcionan (arquitectura y flujo)

- **Captura y registro:** todos los honeypots deben registrar metadatos (timestamp, IP origen, puertos, payloads), y preferentemente capturar tráfico (pcap) o transcript de sesiones.
- **Aislamiento/containment:** los honeypots, especialmente los de alta interacción, se colocan en segmentos aislados o VLANs, con reglas estrictas de salida (firewall, proxy) para evitar que el atacante comprometa otros activos.
- **Tarpitting / sinkholing:** técnicas para retrasar la conexión del atacante (tarpit) o redirigir el tráfico malicioso a una zona controlada (sinkhole).
- **Correlación:** integrar logs y eventos con herramientas de NSM/SIEM para enriquecer la telemetría (Zeek, Suricata, ELK, Security Onion).
- **Análisis automático:** pipelines que extraen muestras (malware), las procesan (antivirus/VM sandbox, YARA, Hashing) y notifican a analistas.

4.5 Implementación práctica y despliegue

- **Diseño de red:** desplegar honeypots en segmentos controlados, preferiblemente con doble interfaz (una para gestión/monitor y otra para red de señuelo).
- **Reglas de salida:** limitar el tráfico saliente desde honeypots de alta interacción para impedir que atacantes lo usen como pivot o plataforma de ataque.
- **Logging centralizado:** forward de logs a un servidor central (ELK/Graylog/SIEM) para correlación y almacenamiento a largo plazo.
- **Automatización:** uso de soluciones como T-Pot o scripts de despliegue (Ansible, Terraform) para provisión reproducible.
- **Actualización y mantenimiento:** revisar periódicamente configuraciones, reglas e indicadores de compromiso (IOCs) detectados.

4.6 Honeytokens (tipos y uso)

- **Cuentas de usuario falsas:** usuarios que nunca deberían iniciar sesión; cualquier intento es evidencia de exploración o credenciales robadas.
- **Documentos-trampa:** ficheros con enlaces o web-beacons únicos; la apertura o click revela exfiltración o movimiento lateral.
- **API keys/Secrets falsos:** si son usados, indican que un atacante encontró o probó credenciales en repositorios o backups.
- **Direcciones de correo trampa:** correos que nunca deberían recibir tráfico; si reciben mensajes, indican que listas han sido expuestas.
- **Registros en bases de datos:** filas con datos sensibles que no deberían ser consultadas; cualquier acceso es indicio de compromiso.

4.7 Buenas prácticas

- **Aislar estrictamente** honeypots de alta interacción con reglas de salida y monitoreo continuo.
- **Instrumentar y centralizar** logs y pcaps para facilitar análisis forense y correlación.
- **Automatizar análisis** de muestras (sandboxing, hashing, YARA) para reducir tiempo de triage.
- **Técnicas de camuflaje:** configurar respuestas realistas para evitar detección por heurísticas simples de fingerprinting.
- **Plan de respuesta:** definir cómo actuar ante capturas relevantes (bloqueo de IPs, actualización de IDS/IPS, notificación a stakeholders, cumplimiento legal).
- **Rotación de honeytokens y credenciales:** evitar que honeytokens válidos se vuelvan obsoletos o comprometan sistemas reales.

4.8 Casos de uso y ejemplos de flujo de trabajo

- **Detección temprana de worms**: un honeypot de baja interacción puede capturar el exploit inicial y las muestras binarios que luego se analizan en sandbox.
- **Inteligencia de amenazas**: correlacionar sesiones SSH capturadas por Cowrie con conexiones de red detectadas por Zeek/Suricata para entender tácticas de acceso.
- **Monitoreo OT/SCADA**: Conpot para identificar escaneos y ataques dirigidos a infraestructuras industriales.
- **Investigación proactiva (honeyclient)**: Thug visitando URLs sospechosas para recolectar payloads de drive-by downloads.

Sistemas que agrupan honeypots

- **T-Pot:** distribución que integra múltiples honeypots y stack de visualización/análisis, útil para despliegues de investigación o labs de monitoreo.

Indicadores de éxito (KPIs)

- Cantidad de muestras de malware recolectadas.
- Número de credenciales honeytoken usadas.
- Tiempo medio desde la interacción hasta el análisis/sandboxing.
- Integración de IOCs con IDS/IPS y bloqueo automático en cortafuegos.

Consideraciones legales y éticas

- Verificar la normativa local e interna sobre captura y almacenamiento de datos de terceros.
- Evitar técnicas de "hacking de retorno" o actos que puedan interpretarse como ofensivos.
- Mantener políticas claras de retención, acceso y divulgación de datos capturados.
- Coordinar con el equipo jurídico y de cumplimiento antes de desplegar honeypots públicos.