Módulo 4.4

Redes y Seguridad
Gr. Ing. Informática
J.L. Vázquez Poletti

Objetivo

En este módulo aprenderemos a usar un IDS muy versátil que puede funcionar en modo Host o Red.

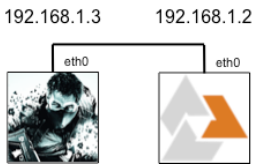
Punto de partida

La empresa tecnológica EUROCORP está desarrollando la versión 6 de su chip DART. Dicha versión viene con una serie de contramedidas pensadas para detectar ataques provenientes de la red en dos modos diferentes. Dichas contramedidas son proporcionadas por un sistema de detección de intrusiones (IDS) llamado **Snort**.

Ha llegado el momento de someter esta funcionalidad a prueba antes de lanzar el chip al mercado y desbancar a la competencia.

Objetivo 1: Escenario

El IDS será probado en modo Host (HIDS) en **RyS-Víctima** mientras que las acciones ofensivas se realizarán por un agente de EUROCORP que ha recibido la actualización **RyS-Backtrack**.



Se pueden abrir varias terminales en las máquinas sin entorno gráfico pulsando **CTRL+ALT+F*N***.

NOTA: Para cambiar el teclado a la disposición española en **RyS-Víctima** hay que ejecutar **loadkeys es** como usuario **root**.

Objetivo 1: Operación básica

En **RyS-Víctima:**

- Activar **snort** para que se limite a escuchar el interfaz activo y muestre mensajes con gran nivel de detalle.

Desde **RyS-Backtrack**:

- Realizar **pings**.

Terminar ambas operaciones tras unos instantes.

IMPORTANTE: Antes de empezar, se deberá apagar el demonio del sistema **snort** (**/etc/init.d/snort stop**).

Es el momento de editar las reglas de **snort** para realizar una detección básica.

Máquinas Virtuales

- RyS-Backtrack (root:toor)
- RyS-Router1 (ubntu:reverse)
- RyS-Víctima (msfadmin:msfadmin)

Más información

Backtrack: http://www.backtrack-linux.org/

Snort: http://www.snort.org/

Wireshark: http://www.wireshark.org/

HPing: http://www.hping.org/

Esto se realiza editando el fichero **/etc/snort/snort.conf**.

1. Mover **snort.conf** a **snort.conf.ORIG**.
2. Borrar **/var/log/snort/alert**.
3. Editar **snort.conf**.
    - Indicar la dirección de **RyS-Víctima**.

      var HOME_NET *<IPVíctima>*

    - Indicar que se desea detectar el tráfico ICMP entrante.

      alert icmp any any -> $HOME_NET any (msg:"Incoming ICMP Traffic";SID:1000001;)

4. Arrancar **snort** (siempre desde **/etc/snort/**) y realizar **ping** desde **RyS-Backtrack**. Acabar con ambos procesos tras unos instantes.
5. Visualizar el contenido de **/var/log/snort/alert**

Ahora se añadirá una regla que alertará de cualquier conexión **telnet** entrante:

```
alert tcp any any -> $HOME_NET 23 (msg:"Incoming TCP Traffic";SID:1000002;)
```

1. Eliminar el registro de alertas y reiniciar **snort**.

   NOTA: El proceso de apagado de **snort** es lento ya que antes realiza una recopilación de estadísticas de tráfico. Si se tarda demasiado en apagar, se puede hacer un apagado drástico con **kill -9**.

2. Realizar una conexión telnet desde **RyS-Backtrack**.

A este punto, se puede ver que las reglas de **snort** tienen la siguiente estructura:

Acción Protocolo IPorigen PuertoOrigen -> IPDestino PuertoDestino (*Parámetros*)

Los parámetros (separados por ";") que se han usado hasta ahora han sido:

- **msg**: mensaje a mostrar en el registro.
- **sid:** identificador de la regla. Las reglas incluídas en **snort** (que no se han activado hasta ahora) van del 100 al 999999. Las siguientes deben superar el límite superior.

Objetivo 1: Contenido de las tramas

Para trabajar con más nivel de detalle hay que activar una serie de módulos específicos llamados preprocesadores y configurarlos en **snort.conf**.

1. Añadir el directorio donde se encuentran los preprocesadores:

   ```
   dynamicpreprocessor directory /usr/lib/snort_dynamicpreprocessor/
   ```

2. Activar el preprocesador **stream5**:

   ```
   preprocessor stream5_global: max_tcp 8192, track_tcp yes, \
   ```

```
track_udp no
```

```
preprocessor stream5_tcp: policy first, use_static_footprint_sizes
```

3. Activar el preprocesador **frag3**:

```
preprocessor frag3_global: max_frags 65536

preprocessor frag3_engine: policy first detect_anomalies
```

4. Activar el preprocesador **flow**:

```
preprocessor flow: stats_interval 0 hash 2
```

Para la siguiente regla, será necesario activar el preprocesador **ftp_telnet**:

```
preprocessor ftp_telnet: global \
  encrypted_traffic yes \
  inspection_type stateful

preprocessor ftp_telnet_protocol: telnet \
  normalize \
  ayt_attack_thresh 200

preprocessor ftp_telnet_protocol: ftp server default \
  def_max_param_len 100 \
  alt_max_param_len 200 { CWD } \
  cmd_validity MODE < char ASBCZ > \
  cmd_validity MDTM < [ date nnnnnnnnnnnnnn[.n[n[n]]] ] string > \
  chk_str_fmt { USER PASS RNFR RNTO SITE MKD } \
  telnet_cmds yes \
  data_chan

preprocessor ftp_telnet_protocol: ftp client default \
  max_resp_len 256 \
  bounce yes \
  telnet_cmds yes
```

5. Crear una regla con las siguientes características:
   - Detecta cuando un usuario ejecuta un comando como usuario root usando el parámetro **Content:“”**.
   - La conexión se ha realizado a la propia máquina por **telnet**.
   - Especificar que la conexión ha sido establecida y en sentido del

servidor usando el preprocesador **flow** (**flow:to_server,established**).

6. Verificar el funcionamiento de la regla iniciando sesión desde **RyS-Backtack**.

Objetivo 1: Escaneo de puertos

Para detectar los escaneos de puertos es necesario el preprocesador **sfportscan**:

```
preprocessor sfportscan: proto  { all } \
        memcap { 10000000 } \
        sense_level { low }
```

1. Iniciar **snort y** ejecutar **nmap** sobre **RyS-Víctima** sin usar más parámetros.
2. Parar **snort**. Leer el archivo de alertas y borrarlo.
3. Reiniciar **snort** y realizar un escaneo **TCP FIN**.
4. Parar **snort**. Leer el archivo de alertas y borrarlo.

¿De dónde sale la regla que ha activado la alerta en ambos escaneos?

5. Iniciar **snort** y realizar cualquier tipo de escaneo pero teniendo como objetivo los puertos de 20 al 30. Parar **snort**, leer el archivo de alertas y borrarlo.
6. Iniciar **snort** y realizar cualquier tipo de escaneo pero teniendo como objetivo los puertos de 20 al 30 pero esta vez fijando un **retardo entre pruebas de 60 segundos**. Parar **snort**, leer el archivo de alertas y borrarlo.

¿Qué es lo que detecta **snort** en ambos casos? ¿Por qué?

Para detectar escaneos el parámetro **flags** es muy interesante ya que permiten filtrar por mensajes TCP:

- **F**: FIN
- **S**: SYN
- **R**: RST
- **P**: PSH
- **A**: ACK
- **U**: URG
- **2**: Bit reservado 2
- **1**: Bit reservado 1

El valor 12 se emplea para detectar intentos de fingerprinting y se suele separar del resto de flags con "," (Ejemplo: AF,12).

Para componer las reglas de los siguientes apartados:

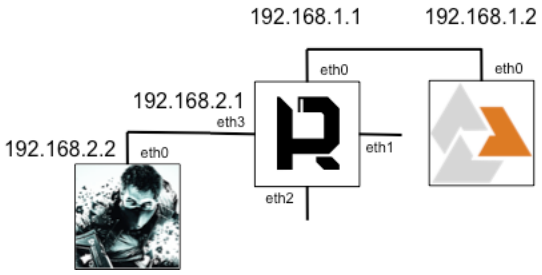
- Usar **wireshark** para capturar el tráfico.
- Realizar un escaneo de prueba dirigido a un solo puerto.

1. Definir una regla que detecte escaneos **TCP FIN**. Probarla frente a otros escaneos.
2. Definir una regla que detecte escaneos **TCP Xmas**. Probarla frente a otros escaneos.
3. Definir una regla que detecte escaneos **TCP NULL**. Probarla frente a otros escaneos.

   PISTA: El parámetro **flags** admite el número de flags presente en la trama.

Objetivo 2: Escenario

El IDS será probado en modo Red (convirtiéndose por tanto en NIDS). Éste será desplegado en un router de la empresa (**RyS-Router1**) que servirá de puerta de enlace para la red de **RyS-Víctima** y la de **RyS-Backtrack**.



- Sólo será necesario reiniciar **RyS-Backtrack** desde **VirtualBox** para acomodarlo a la nueva correspondiente interfaz de red.
- Hay que activar el IP forwarding en **RyS-Router1**.
- **RyS-Router1** es el gateway por defecto de las otras dos máquinas.

Objetivo 2: Configuración NIDS

Ahora se trabajará en **RyS-Router1**, pero antes de empezar hay que preparar el entorno en dicha máquina:

1. Parar el demonio **snort**.
2. Borrar **/var/log/snort/alert**.

Esta vez se usará el fichero por defecto debido a que la versión de **snort** en **RyS-Router1** es mucho más reciente, incorporando nuevas funcionalidades y por tanto aumentando la complejidad de configuración.

Hay que realizar los siguientes cambios en el fichero:

- La red que se desea proteger es **192.168.1.0/24**.
- La red externa es **192.168.2.0/24**.

Ahora **snort** se debe invocar indicando que se desea escuchar la interfaz que le conecta con **RyS-Backtrack** (opción **-i**) y que lea la configuración de **snort.conf** (**-c**).

Objetivo 2: Escaneo de puertos y operaciones básicas

Se comprobarán las funcionalidades de detección de escaneo de puertos en modo NIDS:

1. Activar el preprocesador **sfportscan**.
2. Activar **snort** y realizar un escaneo **TCP FIN**.
3. Realizar un escaneo **TCP Null**.
4. Realizar un escaneo **sin ping**.

¿Qué aspectos del escaneo son detectados en los tres casos?

5. Adaptar la regla para **TCP Null** que se había diseñado en el objetivo anterior.

El análisis del contenido de las tramas se ha mejorado en esta nueva versión.

1. Conectarse por **telnet** pero introducir la contraseña equivocada.
2. Leer el archivo de alertas.

¿Qué aspectos de la conexión son detectados?

Objetivo 2: Detección de ataques

El primer ataque que se detectará es el **TCP SYN Flood**, que ya se vio anteriormente. El programa usado para ello es **hping3** y se pondrá como objetivo el puerto del servidor telnet.

1. Crear una regla que detecte este ataque. Para ello se usará el parámetro **threshold** con las siguientes opciones:
   - type both
   - track by_src
   - count 70
   - seconds 10

¿En qué se basa la detección este ataque?

Objetivo 2: Modo IPS

Snort puede usarse también como sistema de prevención de intrusiones (IPS). Una de las formas es con la acción **reject**, que reemplaza a **alert**.

1. Crear reglas que rechacen los escaneos de puertos **TCP Null, TCP Fin** y **TCP Xmas**.
2. Realizar estos tipos de escaneo.

Se puede usar el parámetro **resp** para cortar las conexiones mediante mensajes **TCP** e **ICMP**:

- **rst_snd**: RST al remitente de la trama que originó la alerta.
- **rst_rcv**: RST al destinatario de la trama que originó la alerta.

- **rst_all**: RST a ambos.
- **icmp_net**: ICMP_NET_UNREACH al remitente.
- **icmp_host**: ICMP_HOST_UNREACH al remitente.
- **icmp_port**: ICMP_PORT_UNREACH al remitente.
- **icmp_all**: las tres respuestas ICMP.

NOTA: se pueden poner varios valores separados por ",".

1. Crear una regla para que se impida conectarse al servidor **ftp** de **RyS-Víctima**.

También se pueden modificar los datos recibidos por un potencial atacante a través del protocolo **http**. Para ello, se emplea el parámetro **react**:

- **block**: bloquea la sesión.
- **warn:** muestra un mensaje de aviso.
- **msg:** define el mensaje que se mostrará con el aviso al usuario.
- **proxy:** muestra el proxy al que se debería conectar el usuario.

1. Bloquear el acceso al servidor web desde la red externa.
2. Bloquear el acceso al servidor web y además mostrar un mensaje de aviso personalizado.