Instalación y uso básico de detectores de intrusiones (Suricata) y herramientas de análisis de vulnerabilidades (OpenVAS)

SSI 2018/19

20 de diciembre de 2018

Índice

1. Entorno de prácticas

1.1. Software de virtualización VIRTUALBOX

En estas prácticas se empleará el software de virtualización VIRTUALBOX para simular los equipos GNU/Linux sobre los que se realizarán las pruebas.

- Página principal: `http://virtualbox.org`
- Más información: `http://es.wikipedia.org/wiki/Virtualbox`

1.2. Imágenes a utilizar

1. Scripts de instalación
   - para GNU/Linux: ejercicio-ids.sh
     `alumno@pc: $ sh ejercicio-ids.sh`
   - para MS windows: ejercicio-ids.ps1
     `Powershell.exe -executionpolicy bypass -file ejercicio-ids.ps1`

   Notas:
   - Se pedirá un identificador (sin espacios) para poder reutilizar las versiones personalizadas de las imágenes creadas (usad por ejemplo el nombre del grupo de prácticas o el login LDAP)
   - En ambos scripts la variable `$DIR_BASE` especifica donde se descargarán las imágenes y se crearán las MVs.
     Por defecto en GNU/Linux será en `$HOME/SSI1819` y en Windows en `C:/SSI1819`.
     Puede modificarse antes de lanzar los scripts para hacer la instalación en otro directorio más conveniente (disco externo, etc)
   - Es posible descargar las imágenes comprimidas manualmente (o intercambiarlas con USB), basta descargar los archivos con extensión `.vdi.zip` de `http://ccia.esei.uvigo.es/docencia/SSI/1819/practicas/` y copiarlos en el directorio anterior (`$DIR_BASE`) para que el script haga el resto.
   - Si no lo hacen desde el script anterior, se pueden arrancar las instancias VIRTUALBOX desde el interfaz gráfico de VirtualBOX o desde la línea de comandos con `VBoxManage startvm <nombre MV>_<id>`
2. Imágenes descargadas
   - **base_ssi.vdi** (1,2 GB comprimida, 4,8 GB descomprimida): Imagen genérica (común a todas las MVs) que contiene las herramientas a utilizar
     Contiene un sistema Debian 9 con herramientas gráficas y un entorno gráfico ligero LXDE (*Lighweight X11 Desktop Environment*) [LXDE].
   - **swap1GB.vdi**: Disco de 1 GB formateado como espacio de intercambio (SWAP)
3. Usuarios configurados e inicio en el sistema
   - Usuarios disponibles

login	password
root	purple
usuario	usuario

   - Acceso al entorno gráfico una vez logueado (necesario para poder copiar y pegar desde/hacia el anfitrión)

     `root@datos:~# startx`

   - Habilitar copiar y pegar desde/hacia el anfitrión en el menú `Dispositivos -> Portapapeles compartido -> bidir`
     de la ventana de la máquina virtual.

1.3. Máquinas virtuales y redes creadas

Redes donde se realizarán los ejercicios:

- Red interna 192.168.100.0/24: máquina **victima** (192.168.100.111), máquina **ids** (192.168.100.111)
- Red externa 193.147.87.0/24: máquina **openvas** (193.147.87.47)

1.4. Pasos previos

- Comprobar el acceso a Internet desde las máquinas **ids** y **openvas** (ajustar la configuración si es necesario)

```
ids:~/# ping www.esei.uvigo.es
openvas:~/# ping www.esei.uvigo.es

(si falla)
ids:~/# ifconfig enp0s3 10.0.2.15/24
ids:~/# route add default gw 10.0.2.2 dev enp0s3
ids:~/# echo "nameserver 8.8.8.8" > /etc/resolv.conf

openvas:~/# ifconfig enp0s3 10.0.2.15/24
openvas:~/# route add default gw 10.0.2.2 dev enp0s3
openvas:~/# echo "nameserver 8.8.8.8" > /etc/resolv.conf
```

- Forzar las rutas para que el tráfico entre las máquinas **openvas** y **victima** sea posible

```
openvas:~/# route add -net 192.168.100.0/24 dev enp0s8

victima:~/# route add -net 193.147.87.0/24 dev enp0s8
```

  Comprobar la conectividad entre ambas máquinas con `ping`

  Nota: Esta configuración no tiene sentido en un escenario real.

  - El establecimiento manual de estas rutas en este ejemplo evita tener que contar y configurar un router/firewall entre ambas subredes.

2. Instalación y configuración del IDS Suricata

2.1. Suricata IDS

Web Suricata: `https://suricata-ids.org/`

- Documentación: `https://suricata.readthedocs.io/en/suricata-4.1.0/`
- Interfaces gráficos:
  - Visualización de logs: `https://evebox.org/`
  - Gestión de reglas:`https://github.com/StamusNetworks/scirius`
- Distribuciones que lo incluyen
  - SimpleWall: `http://www.simplewallsoftware.com/`
  - Security Onion: `https://securityonion.net/` `https://blog.securityonion.net/`
  - SELKS: `https://www.stamus-networks.com/open-source/#selks`

Web SNORT (otro IDS Open Source): `https://www.snort.org/`

2.2. Pasos a seguir

1. Habilitar el repositiorio *backports* de Debian 9, para instalar la última versión disponible de Suricata (v 4.0)

```
ids:~/# echo "deb http://ftp.debian.org/debian stretch-backports main" >> /etc/apt/sources.list
```

2. Actualizar la lista de paquetes e instalar `suricata` y

```
ids:~/# apt-get update
ids:~/# apt-get install suricata suricata-oinkmaster -t stretch-backports
```

3. Forzar la descarga de las reglas libres de `https://rules.emergingthreats.net/`

```
ids:~# suricata-oinkmaster-updater
```

   - Descarga la última versión de las reglas desde https://rules.emergingthreats.net/open/suricata-4.0.0/emerging.rules.ta [ver configuración en `/etc/suricata/suricata-oinkmaster.conf`]
   - Las reglas descargadas se ubicarán en `/etc/suricata/rules` y se registrarán en el fichero de configuración `/etc/suricata/suricata.yaml`

4. Ajustar la configuración de Suricata al escenario planteado.

```
ids:~# cd /etc/suricata
ids:/etc/suricata# nano suricata.yaml
```

   - Establecer la dirección de la red monitorizada (`192.168.100.0/24`)

```
...
##
## Step 1: inform Suricata about your network
##

vars:
  # more specifc is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.100.0/24]"        <----- CAMBIAR  AQUI

    EXTERNAL_NET: "!$HOME_NET"
...
```

   - Identificar las reglas cargadas (de momento no se cambiará nada)

```
...
##
## Step 2: select the rules to enable or disable
##

default-rule-path: /etc/suricata/rules
rule-files:
 - botcc.rules
 # - botcc.portgrouped.rules
 - ciarmy.rules
 - compromised.rules
 ...
 - smtp-events.rules    # available in suricata sources under rules dir
 - dns-events.rules     # available in suricata sources under rules dir
 - tls-events.rules     # available in suricata sources under rules dir
 # - modbus-events.rules  # available in suricata sources under rules dir
 # - app-layer-events.rules  # available in suricata sources under rules dir
 # - dnp3-events.rules       # available in suricata sources under rules dir
 # - ntp-events.rules       # available in suricata sources under rules dir

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

   - Identificar los tipos de salida soportados (de momento no se cambiará nada)

```
...
##
## Step 3: select outputs to enable
```

```
##

default-log-dir: /var/log/suricata/

# global stats configuration
stats:
  enabled: yes
  interval: 8

# Configure the type of alert (and other) logging you would like.
outputs:
  # a line based alerts log similar to Snort's fast.log
  - fast:
      enabled: yes
      filename: fast.log
      append: yes

  # Extensible Event Format (nicknamed EVE) event log in JSON format
  - eve-log:
      enabled: yes
      filetype: regular #regular|syslog|unix_dgram|unix_stream|redis
      filename: eve.json
      ...
      types:
        - alert:
            # payload: yes             # enable dumping payload in Base64
            ...
            metadata: yes              # add L7/applayer fields, flowbit and ot$
            ...
        - http:
            extended: yes     # enable this for extended logging information
            ...
        - dns:
            query: yes     # enable logging of DNS queries
            answer: yes    # enable logging of DNS answers
        - tls:
            extended: yes     # enable this for extended logging information
    ...
...
```

Define el directorio donde se ubicará la salida del detector de intrusiones (`/var/log/suricata`)

La configuración por defecto define 3 tipos de salidas

- `suricata.log`: log del demonio Suricata, con las acciones realizadas por el IDS y los errores que se hayan producido (útil en la depuración de las reglas)
- `eve.josn`: salida de Suricata con las alertas detectadas y detalles del tráfico capturado (en formato JSON, típicamente se usa como entrada a herramientas de visualización de logs)
- `fast.log`: salida abreviada con información de las alertas generadas (campos `msg` y `classtype` de las reglas)

Es posible habilitar otros tipos de salida, con información de protocolos específicos (DNS, HTTP, TLS, etc). Ver sección `## Step 5: App Layer Protocol Configuration`

- Establecer la tarjeta de red desde la cuál se capturará el tráfico (`enp0s8`)

```
...
##
## Step 4: configure common capture settings
##
## See "Advanced Capture Options" below for more options, including NETMAP
## and PF_RING.
##

# Linux high speed capture support
```

```
af-packet:
  - interface: enp0s8            <----- CAMBIAR  AQUI
  ...
...
```

5. Reiniciar el demonio Suricata

```
ids:/etc/suricata# systemctl restart suricata.service
```

3. Comprobación del funcionamiento del IDS

Previo: Para visualizar las alertas del fichero `eve.json` se usará la herramienta `jq`, un *pretty printer* para datos en formato JSON

```
ids:~# apt-get install jq
```

3.1. Detección de PING y escaneo de puertos

1. Lanzar el comando Ping desde la máquina `openvas` hacia la máquina `victima`

```
openvas~# ping victima.ssi.net
```

2. Lanzar un escaneo de puertos completo con NMAP desde la máquina `openvas` hacia la máquina `victima`

```
openvas~# nmap -sV -O  victima.ssi.net
```

3. Comprobar las alertas en los ficheros de log de Suricata en la máquina `ids`

```
ids~# tail -f /var/log/suricata/fast.log

ids~# tail -f /var/log/suricata/eve.json

(MEJORA: filtradas/visualizadas con jq)
ids~# tail -f /var/log/suricata/eve.json | jq  'select(.event_type=="alert")'
```

3.2. Ejemplo de creación de reglas propias

1. Ejemplo con tráfico UDP (sin conexión)
   Se usará la librería Pyhton Scapy para generar paquetes que fuercen la activación de las alertas definidas.
   - Web de SCAPY : http://www.secdev.org/projects/scapy/

   a) Añadir a la configuración de Suricata la carga de un fichero `local.rules` con reglas propias

```
ids:/etc/suricatas# nano /etc/suricata/suricata.yaml

...
##
## Step 2: select the rules to enable or disable
##

default-rule-path: /etc/suricata/rules
rule-files:
  - local.rules                  <----- AÑADIR  AQUI
  - botcc.rules
  # - botcc.portgrouped.rules
  - ciarmy.rules
  - compromised.rules
...
```

b) Crear una regla Snort para capturar un paquete UDP con un *Payload* concreto.

```
ids:/etc/suricatas# nano /etc/suricata/rules/local.rules

alert udp $HOME_NET any -> $EXTERNAL_NET 3333 (
                          msg: "Prueba SSI con UDP";
                          content: "ho ho ho";
                          flow:to_server;
                          nocase;
                          sid:9000001;)
```

Importante: todos los elementos de la regla deben de ir en una única línea
Más detalles sobre formato de reglas SNORT/Suricata:

- `https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricata_Rules`
- `https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Flow-keywords`
- `https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Payload_keywords`

c) Reiniciar Suricata

```
ids:/etc/suricata# systemctl restart suricata.service
```

d) Desde la máquina `victima` usar Scapy para generar un paquete UDP que encaje con la regla creada

```
victima:~# scapy

>>> ip=IP()
>>> ip.src="192.168.100.111"
>>> ip.dst="193.147.87.47"

>>> udp=UDP()
>>> udp.dport=3333
>>> udp.sport=11111

>>> payload="ho ho ho los reyes magos son los padres"

>>> paquete = ip/udp/payload
>>> paquete.show()

>>> send(paquete)

>>> exit()
```

e) Verificar en `fast.log` y `eve.json` de la máquina `ids` la captura del tráfico generado
Nota: La actualización de ambos ficheros no es inmediata.

- Puede forzarse la escritura de los ficheros reiniciando el demonio de Suricata con `systemctl restart suricata.s`

2. Ejemplo con tráfico TCP (con conexión)

   Uso de `nc` (*netcat*) para simular servidores y clientes.

   - En el caso de conexiones TCP, las reglas con el valor `established` en el parámetro `flow` requieren completar la negociación en 3 pasos de una conexión TCP.
   - Además, se requiere contar con un servidor escuchando en los puertos implicados en la regla.
   - Es posible hacerlo con Scapy, pero suele ser más sencillo simular un servidor y su cliente con la herramienta `netcat` (comando `nc`)

   a) Añadir una nueva regla en `local.rules`

```
ids:/etc/suricatas# nano /etc/suricata/rules/local.rules

...
alert tcp $EXTERNAL_NET any -> $HOME_NET 4444 (
                                    msg:"prueba SSI tcp";
                                    content: "feliz navidad";
                                    flow:to_server,established;
                                    nocase;
                                    sid:9000002;)
```

b) Reniciar Suricata

```
ids:/etc/suricata# systemctl restart suricata.service
```

Importante: todos los elementos de la regla deben de ir en una única línea

c) Simular un servidor en `victima` y un cliente en `openvas` que genere tráfico para activar la regla anterior.

```
victima:~# nc -l -p 4444


openvas:~# nc victima.ssi.net 4444
               (escribir en el terminal feliz navidad)
```

d) Verificar en `fast.log` y `eve.json` de la máquina `ids` la captura del tráfico generado

3.3. Generación de tráfico para reglas EemergingThreats

Descarga de reglas: `https://rules.emergingthreats.net/`

3.3.1. Reglas emerging-user_agents.rules

Capturan peticiones HTTP con valores `UserAgent` sospechosos.

1. Revisar fichero de reglas `/etc/suricata/rules/emerging-user_agents.rules`

```
ids:~# leafpad /etc/suricata/rules/emerging-user_agents.rules


...
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ET USER_AGENTS Suspicious user agent (Google page)";
                          flow:to_server,established; content:"Google page"; depth:11; http_user_agent;
                          classtype:trojan-activity; sid:2017067; rev:5;
                          metadata:created_at 2011_05_31, updated_at 2011_05_31;)
...
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ET USER_AGENTS Suspicious User-Agent (Windows 8)";
                          flow:to_server,established; content:"Windows 8"; depth:9; http_user_agent;
                          metadata: former_category USER_AGENTS; classtype:bad-unknown; sid:2026520; rev:1;
                          ...
                          updated_at 2018_10_18;)
...
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ET USER_AGENTS VPNFilter Related UA (Gemini/2.0)";
                          flow:established,to_server; content:"Gemini/2.0"; http_user_agent; depth:10;
                          fast_pattern; isdataat:!1,relative; metadata: former_category USER_AGENTS;
                          reference:url,twitter.com/m0rb/status/1021626709307805696;
                          classtype:trojan-activity; sid:2025889; rev:1; ...;)
...
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ET USER_AGENTS Suspicious User-Agent Detected (RLMultySocket
                          flow:established,to_server; content:"RLMultySocket"; http_user_agent; depth:13;
                          isdataat:!1,relative; threshold:type limit,count 2,track by_src,seconds 300;
                          metadata: former_category TROJAN; reference:url,doc.emergingthreats.net/bin/view/Main/
                          classtype:trojan-activity; sid:2008603; rev:9; ...;)...
```

2. Generar peticiones HTTP usando CURL desde la máquina `vicitima` hacia `openvas` con los valores `UserAgent` sospechosos (parámetro `-A`).

```
victima:~# curl -A "Google page"   openvas.ssi.net
victima:~# curl -A "Windows 8"     openvas.ssi.net
victima:~# curl -A "Gemini/2.0"    openvas.ssi.net
victima:~# curl -A "RLMultySocket" openvas.ssi.net
```

Nota 1: Asegurar que en la máquina `openvas` está en ejecución el servidor web Apache (y reniciarlo si es necesario)

```
openvas:~# systemctl status apache2.service

openvas:~# systemctl restart apache2.service   (si Apache estaba parado)
```

Nota 2: También es posible detener el servidor web Apache poner el comando `nc` en modo escucha en el puerto 80

```
openvas:~# systemctl stop apache2.service

openvas:~# nc -l -p 80
```

3. Verificar en `fast.log` y `eve.json` de la máquina `ids` la captura del tráfico generado

3.3.2. Reglas con contenido binario

1. Reglas para detectar el envío de código de exploits

```
ids:~# leafpad /etc/suricata/rules/emerging-exploit.rules

...
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 1433 (msg:"ET EXPLOIT MS-SQL SQL Injection closing string plus line
                              flow: to_server,established;
                              content:"'|00|"; content:"-|00|-|00|";
                              reference:url,owasp.org/index.php/SQL_Injection; reference:url,doc.emergingthreats
                              classtype:attempted-user; sid:2000488; rev:7;...;)
...



victima:~# nc -l -p 1433

openvas:~# echo -e "'\x00     -\x00-\x00" | nc victima.ssi.net 1433
```

2. Reglas para servidores de BD

```
ids:~# leafpad /etc/suricata/rules/emerging-sql.rules
...
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 3306 (msg:"GPL SQL MYSQL root login attempt";
                              flow:to_server,established;
                              content:"|0A 00 00 01 85 04 00 00 80|root|00|";
                              classtype:protocol-command-decode;
                              sid:2101775; rev:4; metadata:created_at 2010_09_23, updated_at 2010_09_23;)
...

victima:~# systemctl stop mysqld.service
victima:~# nc -l -p  3306

openvas:~# echo -e "\x0A\x00\x00\x01\x85\x04\x00\x00\x80root\x00" | nc victima.ssi.net 3306
```

3. Verificar en `fast.log` y `eve.json` de la máquina `ids` la captura del tráfico generado

3.3.3. Habilitar ficheros de reglas

1. Habilitar (descomentar) reglas `emerging-shellcode.rules` en `suricata.yaml` y reiniciar IDS.

```
ids:~# leafpad /etc/suricata/suricata.yaml

...
##
## Step 2: select the rules to enable or disable
##
```

```
default-rule-path: /etc/suricata/rules
rule-files:
 - local.rules
 - botcc.rules
 ...
 - emerging-scan.rules
 - emerging-shellcode.rules       <--- descomentar aqui
 - emerging-smtp.rules
 - emerging-snmp.rules
 - emerging-sql.rules
 ...
...

ids:/etc/suricata# systemctl restart suricata.service
```

2. Verificar regla a activar y generar tráfico

```
ids:~# leafpad /etc/suricata/rules/emerging-shellcode.rules
...
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ET SHELLCODE Possible Call with No Offset TCP Shellcode";
                          flow:established;
                          content:"|E8 00 00 00 00 0F 1A|";
                          reference:url,www.networkforensics.com/2010/05/16/network-detection-of-x86-buffer-ove
                          classtype:shellcode-detect; sid:2012090; ...;)

victima:~# nc -l -p 9999

openvas:~# echo -e "\xE8\x00\x00\x00\x00\x0F\x1A" | nc victima.ssi.net 9999
```

3. Verificar en `fast.log` y `eve.json` de la máquina `ids` la captura del tráfico generado

4. TAREA ENTREGABLE

4.1. Tareas a realizar

1. Seleccionar tres reglas (ver `/etc/suricata/rules/`) y generar una serie de paquetes que fuercen su activación (con Scapy, con `nc` o directamente con los servidores y clientes adecuados)
   - Resumen de las reglas SNORT/Suricata
   - Documentación reglas SNORT/Suricata (ver capítulo 3)
   - Más detalles sobre formato de reglas SNORT/Suricata:
     - Estructura de las reglas: `https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricata_Rules`
     - Flujo de tráfico: `https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Flow-keywords`
     - Payload: `https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Payload_keywords`
2. Comprobar la detección de ese tipo de tráfico por parte de Suricata en sus ficheros de log
3. **Importante:** al menos una de las reglas seleccionadas debe requerir tráfico en binario

4.2. Documentación a entregar

- Documentar los intentos de generación de alertas SNORT/Suricata realizados en la sección 4.1
  - Incluir cada una de las tres reglas seleccionadas y describir textualmente en qué consisten (servicios/aplicaciones implicados, vulnerabilidad concreta, contenidos/condiciones de la regla, ...)

    - Describir el procedimiento seguido para dispararlas
    - Comentar el resultado: éxito (mostrando capturas de los logs generados) o fracaso
- Fecha límite: 8/1/2018 (día del examen)

5. Ejemplo EXTRA: Instalación y uso del escaner de vulnerabilidades OpenVAS (no entregable)

Web OpenVAS: http://www.openvas.org/

`<pendiente>`