


Criterios de evaluación y condiciones de entrega

- Para cualquier duda y/o aclaración sobre el enunciado tenéis que dirigiros al consultor responsable de vuestra aula.
- Se tiene que entregar la solución en un archivo, preferiblemente, en formato PDF .
- El nombre del archivo tiene que ser `ApellidoNombre_asignatura_LAB1` con extensión `.pdf`
- Para el segundo ejercicio deberéis también entregar el archivo con nombre `ApellidoNombre_asignatura_LAB1_rule.snort`
- Razonad la respuesta en todos los ejercicios e indicad todos los pasos que habéis realizado para obtener la solución.
- Las respuestas sin justificación, que sean una copia de una fuente de información y/o que no contengan las referencias utilizadas, no recibirán ninguna puntuación.
- La fecha límite de entrega será el día **2 de diciembre del 2019** antes de las **23:59h**. La entrega se realizará a través del REC (*Registro de Evaluación Continuada*).

Enunciado

Recientemente varias compañías se han puesto en contacto con el CSIRT (*Computer Security Incident Response Team*) de la empresa de seguridad para la que trabajáis. Dichas empresas forman parte del conjunto de clientes a los cuales ofrecéis diversos servicios de seguridad. Estas empresas han solicitado realizar un análisis de ingeniería inversa sobre un conjunto de archivos PDF sospechosos. Los archivos han sido recibidos por diversos usuarios de las empresas vía correo electrónico. Aparentemente, todas ellas han sufrido incidentes de seguridad con características similares tras la apertura de tales archivos. La primera hipótesis apunta a que sendos archivos explotan una vulnerabilidad de tipo *0-day* sobre un lector de archivos PDF muy común, y que ésto ha desencadenado los incidentes posteriores. La sospecha se hace más evidente al verificar que todos los usuarios que recibieron los emails usan el mismo lector de archivos PDF.

El departamento responsable de realizar la ingeniería inversa lleva trabajando toda la noche analizando los archivos PDF. Su objetivo es intentar determinar qué vulnerabilidad explotan, así como esclarecer su finalidad. Un primer análisis estático revela que los archivos PDF contienen *malware* incrustado que posiblemente es ejecutado gracias a la vulnerabilidad *0-day*. Tras extraer y aislar el *malware* de los PDF, se procede a su análisis mediante la herramienta *IDA*. A pesar de los esfuerzos realizados por los analistas, las técnicas de polimorfismo, *anti-disassembly* y *anti-debugging* incluidas en el *malware* hacen de este proceso una tarea lenta y ardua. Durante la noche, uno de los últimos fichajes en el departamento – y antiguo alumno del MISTIC – sugiere realizar un análisis dinámico. En concreto, propone analizar si el *malware* incrustado en los archivos PDF genera algún tipo de tráfico de red. El análisis revela que el *malware* incluido en los archivos PDF se trata, en realidad, de un *downloader*.

Con el objetivo de determinar los sistemas infectados, y mientras que no se genera un parche que solucione la vulnerabilidad, se os propone escribir **una única regla** para *Snort* que detecte la presencia del *downloader* a partir del tráfico de red. La finalidad es desplegar dicha regla en los NIDS de vuestras empresas cliente. Para esto, se os proporcionan varios ficheros que contienen el tráfico de red generado

por seis variantes encontradas del *downloader*. Estos ficheros están en formato PCAP (*Packet CAPture*), el cual es ampliamente utilizado por diversas herramientas de análisis y manipulación de paquetes de red.

Los archivos mencionados, sus valores *hash* SHA256, y sus enlaces de descarga son los siguientes:

# Muestra	Valor SHA256	Link
1	b2ac93e5869a49005db1d3180f20b31e2226c3ae512618103e8670889cc2738b	↓
2	d79ce8334a697c38f50794fe5a7a3a95d10c9e9b32eb45f42e7a3ccd6a5af98f	↓
3	8ababde23ee6b8d26e07332f752d5618b38a5c568ca9e3b40e5dea5d07110d30	↓
4	cfd4e2765ecc0f467926f3355a4335a0225cf7bcd45bab6778ec38f3a52e5be	↓
5	0669c7de94c054182712a4cb9994b3a78fc4a48f9dc29b4e11198fd89f4d6f14	↓
6	f75428cbe1cd5fcfae9042112e35885adc8efd37d8abe175e381424e16160616	↓

Cuadro 1: Archivos .pcap capturados de las diferentes variantes

Teniendo en cuenta el enunciado descrito anteriormente, resolved cada uno de los siguientes ejercicios y responded a las preguntas que se formulan. Recordad que debéis razonar las respuestas e indicar todos los pasos que habéis realizado para obtener la solución.

Para los ejercicios 2 y 3 deberéis instalaros el software de virtualización *VirtualBox* (<https://www.virtualbox.org/>) y luego crear con éste una máquina virtual de tipo GNU/Linux. Configurad dicha máquina virtual con 512MB de RAM y usando como disco duro virtual el archivo *mistic.vdi*. Dicho archivo lo obtendréis tras descomprimir el fichero que está disponible en la URL <http://deic.uab.es/~scastillo/mistic.zip>. En esta máquina virtual ya encontrareis instalado *Snort* y diversos editores de texto. Utilizad el usuario *root* para realizar los ejercicios, y cuya contraseña es *mistic2019*. En el *\$HOME* de dicho usuario existe el directorio *pcaps* donde ya están disponibles los archivos de las capturas.

Ejercicio 1 (2 puntos)

Instalad en vuestro sistema el analizador de paquetes *Wireshark* que podréis obtener de <http://www.wireshark.org/>, y descargad los archivos .pcap mencionados anteriormente. Abrid con dicha herramienta cada uno de los archivos y responded a las siguientes preguntas:

- Buscad en Internet y definid cuál es la finalidad del *malware* de tipo *downloader* ¿Concuerda aparentemente su definición con la secuencia y el contenido de los paquetes que muestra *Wireshark* con cada una de las variantes? ¿Por qué?
- ¿Qué protocolo/s de la capa de aplicación utiliza el *downloader*?
- A partir de las direcciones IP a las que se conecta el *downloader* para realizar las descargas, y utilizando el protocolo *whois*, podéis decir cuál podría ser la procedencia geográfica del *malware*? Para realizar este apartado podéis emplear el comando *whois* de los sistemas *NIX, o bien servicios online como por ejemplo <http://whois.domaintools.com>.

Ejercicio 2 (2 puntos)

Escribid una única regla de tipo *alert* para *Snort* que permita detectar las variantes de *malware* encontradas, y cualquier otra potencialmente futura. Para esto, basaros exclusivamente en los cinco

primeros archivos `.pcap`, e ignorad el sexto que lo usaremos para el siguiente ejercicio. Os recomendamos que consultéis el manual de usuario de *Snort* que encontraréis en <https://www.snort.org/documents>; en concreto, os será de utilidad el capítulo número tres. Tened en cuenta que la regla debe cumplir las siguientes restricciones:

- Para la definición, ignorad las resoluciones DNS y centraros en las cabeceras del protocolo de capa de aplicación que el *downloader* usa para las descargas.
- La regla debe emplear expresiones regulares para contemplar las variantes.
- La regla debe definirse con los siguientes valores fijados:
 - Mensaje de alerta: “*MISTIC malware trojan detected*”.
 - Valor de `sid`: 124444.
 - Valor de revisión: 1.
 - Valor de prioridad: 4.
 - Valor de `classtype`: “*trojan-activity*”.
- Debéis verificar el correcto funcionamiento de la regla con los cinco archivos. Para tal fin, utilizad alguno de los parámetros de *Snort* como son `-r`, `--pcap-list=`, `--pcap-file=`, o bien `--pcap-dir=`.
- Para verificar el funcionamiento de la nueva regla, añadidla al final del archivo de configuración `/etc/snort/snort.conf`.

Una vez definida la regla, justificad cada uno de los campos de la misma. Cread un archivo con el formato `ApellidoNombre_asignatura_LAB1_rule.snort` que **contenga solamente la regla** y entregadlo a través del REC (*Registro de Evaluación Continuada*).

Ejercicio 3 (2 puntos)

La muestra de tráfico capturada número seis tiene una serie de particularidades que la hacen especial en comparación al resto. A pesar de estas particularidades, nos han asegurado que el tráfico generado proviene de una de las variantes del *downloader*. Utilizad *Snort* para evaluar el archivo `.pcap` con vuestra regla y responded a las siguientes preguntas:

- ¿Ha detectado *Snort* la intrusión con vuestra regla?
- ¿Por qué creéis que no ha sido detectada?
- En cualquier caso, ¿Muestra *Snort* algún tipo de alerta sospechosa?
- ¿Qué técnica ha usado el *downloader* para intentar evadir la detección?

Para dar respuesta a las preguntas anteriores inspeccionad el archivo detenidamente con *Wireshark* y leed el artículo que lleva por título “*Evading NIDS, revisited*” de Sumit Siddharth; disponible en la URL <http://www.symantec.com/connect/articles/evading-nids-revisited>.

Sabiendo que las máquinas clientes infectadas son sistemas GNU/Linux, modificad la configuración del preprocesador *frag3* en el archivo `/etc/snort/snort.conf`. En concreto, estableced en la directiva *frag3_engine* la política a *policy linux* en lugar de *policy windows*. Ejecutad de nuevo *Snort* y verificad que ahora el *downloader* sí es detectado. Responded ahora a las siguientes preguntas:

- ¿Por qué ahora es posible detectarlo?
- ¿Cuál es la funcionalidad del preprocesador *frag3*?
- ¿Qué implica la modificación de la configuración *policy linux* en la directiva *frag3_engine*?
- ¿Por qué existen diferentes políticas para *frag3_engine* según el sistema operativo? Consultad la sección 2.2.1 del manual de usuario de *Snort*.

Ejercicio 4 (2 puntos)

Dada la siguiente regla de la base de datos de *Snort*:

```

alert tcp $EXTERNAL_NET any -> $HOME_NET 515
(msg:"EXPLOIT Redhat 7.0 lprd overflow";
flow:to_server,established;
content:"XXX%.172u%300|24|n";
reference:bugtraq,1712; reference:cve,2000-0917;
classtype:attempted-admin; sid:302; rev:9;)

```

Describid qué detecta y a que se corresponden cada uno de los campos de la regla.

Ejercicio 5 (2 puntos)

Dado el siguiente *script* que añade reglas al sistema de *firewalling* de GNU/Linux denominado *iptables*:

```

#!/bin/bash
iptables -F
iptables -X

iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
iptables -N WHITELIST

iptables -A INPUT -i lo -j ACCEPT

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp --dport ssh -j WHITELIST
iptables -A INPUT -p tcp --dport ssh -m state --state NEW -m recent --set
iptables -A INPUT -m recent --update --seconds 30 --hitcount 6 -j LOG
iptables -A INPUT -m recent --update --seconds 30 --hitcount 6 -j DROP
iptables -A INPUT -p tcp --dport ssh -m state --state NEW -j ACCEPT

iptables -A WHITELIST -s 10.0.0.0/16 -j RETURN
iptables -A WHITELIST -s 10.1.0.0/16 -j RETURN
iptables -A WHITELIST -s 10.2.0.0/16 -j RETURN
iptables -A WHITELIST -s 10.3.0.0/16 -j RETURN
iptables -A WHITELIST -j DROP

iptables -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

```

Consultad la documentación que podéis encontrar en las URLs <http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html> y <http://ipset.netfilter.org/iptables.man.html> y resolved cada uno de los siguientes puntos:

- Explicad en términos generales cuál es el comportamiento de un *firewall* que tiene las reglas recogidas en el *script* anterior.
- Explicad cada uno de los comandos de *iptables* que aparecen en el *script* y justificad porqué están en ese orden.
- ¿Cual sería el comando que permitiría registrar las conexiones SSH provenientes de IPs no autorizadas (que no están en la lista blanca)? ¿En que posición dentro del *script* añadiríais dicho comando?

NOTA: como ayuda para comprender mejor el funcionamiento de las reglas, podéis clonar la máquina virtual creada en el ejercicio 2 y asignarle una nueva IP dentro de vuestra red virtual. A partir de esto, podéis experimentar con el *script* anterior modificándolo, y lanzando conexiones de una máquina virtual a la otra para comprobar cual es el comportamiento del filtrado.