Seguridad en redes

# Práctica 1

Pablo Riutort Grande

9 de diciembre de 2019

# 1.

## 1.1.

Los *downloaders* son un tipo de malware especializado cuya funcionalidad es la de descargarse contenido. Se trata de un malware de bajo coste, poco riesgo, reusable y bastante común.
El contenido de la descarga suele ser variable: comandos, archivos de configuración, aplicaciones, upgrades, etc.
Las pequeñas descargas son más difíciles de detectar que si se tratase de una descarga más grande y, además, reduce el riesgo de fallo. Si fuese detectado por un antivirus, el atacante puede crear otros componentes del *downloader* y reutilizar gran parte de los recursos.
El alcance del daño está solo limitado a lo que el malware se pueda descargar.

En las capturas de Wireshark podemos observar que siempre se hace una petición a un archivo "update.zip".


Figura 1: Petición de tipo GET del protocolo HTTP a un archivo update.zip

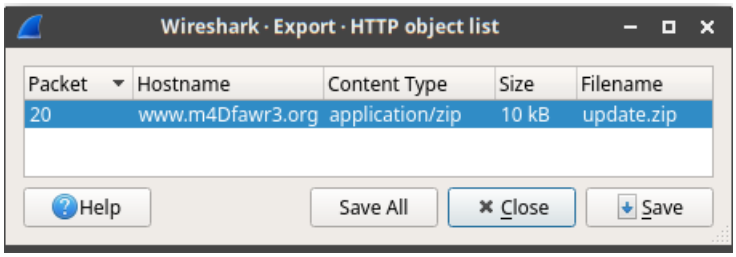
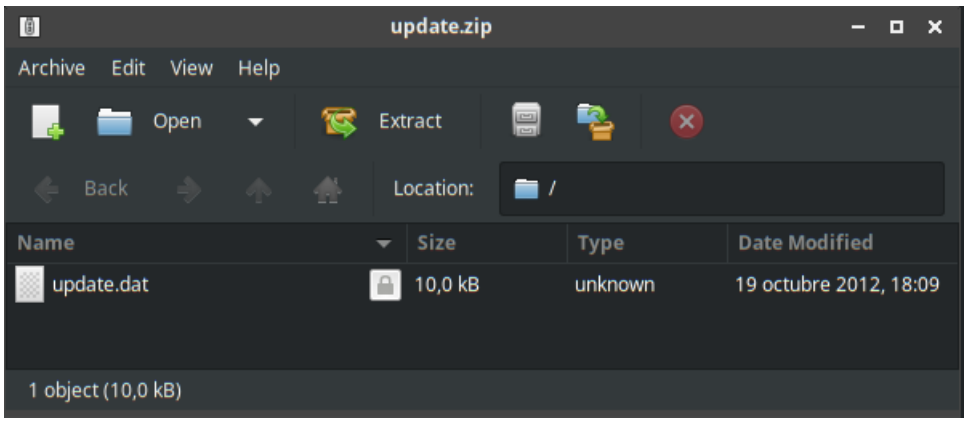



Figura 2: Exportación de objeto HTTP

Si descargamos este archivo veremos que se trata de un pequeño archivo de 10kB protegido con contraseña.



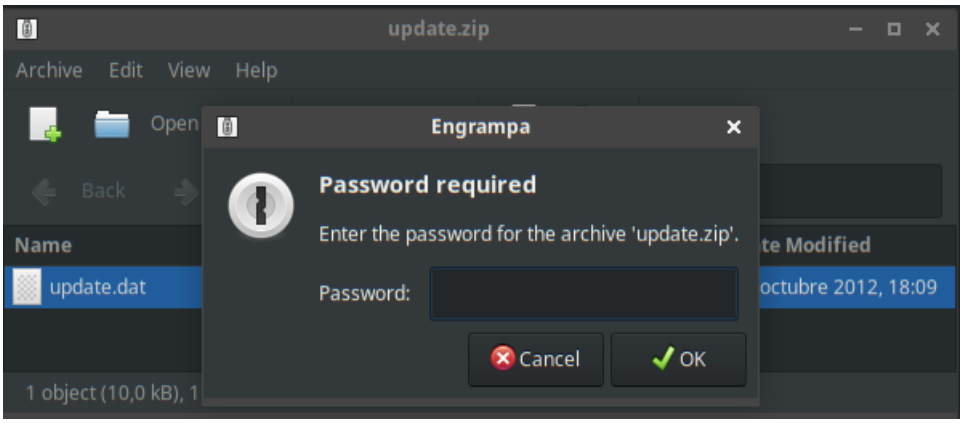



Figura 3: Contenido del archivo update.zip

Aparentemente, la descripción del dowloader concuerda con la secuencia y contenido de los paquetes, puesto que se ha descargado un archivo de unos pocos kilobytes de naturaleza sospechosa: llamado "update" y protegido por contraseña.

## 1.2.

Los protocolos correspondientes a la capa de aplicación utilizados por el *downloader* son DNS y HTTP.

```
10.0.0.2        10.0.0.1        DNS     127 Standard query response
10.0.0.1        58.14.0.1       HTTP    187 GET /update.zip HTTP/1.1
```

Figura 4: Captura de Wireshark mostrando IPs y protocolos implicados

## 1.3.

Para este apartado se ha ejecutado el comando *whois* sobre las direcciones IP referenciadas por el *downloader*:

- 58.14.0.1

```
inetnum:        58.14.0.0 - 58.15.255.255
netname:        JNGDNET
country:        CN
descr:          Jinan Radio \&TV Wellunited
descr:          Digital Cable TV Co., Ltd
descr:          3F Sanjianyinyuan Building,No.22 minsheng Road
descr:          Jinan, Shandong, P.R.China
...
address:        Beijing, China
```

- 61.29.128.91

```
inetnum:        61.29.128.0 - 61.29.159.255
netname:        CHINANETCENTER
descr:          ChinaNetCenter Ltd.
```

```
country:        CN
...
address:        Beijing, China
```

- 123.64.99.200

```
inetnum:        123.64.0.0 - 123.95.255.255
netname:        CTTNET
descr:          China TieTong Telecommunications Corporation
descr:          Jinze Mansion, 2 Guangningbo Street,
descr:          Xicheng District, Beijing, China, 100032
country:        CN
...
address:        Beijing, China
```

- 166.111.111.166

```
inetnum:        166.111.0.0 - 166.111.255.255
netname:        TUNET
descr:          imported inetnum object for IIINT
country:        CN
...
address:        Room 224, Main Building
                Tsinghua University
                Beijing, 100084
```

- 203.100.192.58

```
inetnum:        203.100.192.0 - 203.100.207.255
netname:        ZLLX
descr:          Beijing Zhonglianlixin Technology Co., Ltd.
admin-c:        LH2804-AP
tech-c:         LH2804-AP
country:        CN
...
address:        3F,Building39,Shaoyaoju,Chaoyang District,
    Beijing, China
```

- 210.14.128.7

```
inetnum:        210.14.128.0 - 210.14.143.255
netname:        SHUJUJIA
descr:          Beijing ShuJuJia Technology Co., Ltd.
descr:          Triumph 170 Kai Xuan Cheng, 26th Floor, Block
    C
descr:          Bei Yuan Road, Chaoyang District, Beijing City
country:        CN
...
address:        Beijing CNISP Technology Co., Ltd
```

La procedencia geográfica del malware es de **Beijing, China**.

# 2.

## 2.1.

La regla definida para Snort es la siguiente:

```
alert tcp 10.0.0.1 any -> any any (msg:"MISTIC malware trojan
    detected";sid:124444;rev:1;priority:4;classtype:trojan-activity
    ;flow:to_server,established;content:"User-Agent";http_header;
    pcre:"/^User-Agent\s*:[^\n]*malw4r3/";)
```

Figura 5: Regla definida para detectar *downloader* con Snort

- **alert**. Snort puede efectuar varias acciones si se cumple la regla definida: En este caso, generará una acción de tipo "alert".
- **tcp**. Tipo de paquete sobre el que debe actuar esta regla. En este caso, sobre TCP.
- **10.0.0.1 any → any any**. Aquí se define el origen, destino y dirección del paquete. En este caso tenemos como IP de origen el de las pcaps: 10.0.0.1. Como IP de destino, puertos de origen y destino pueden ser cualquiera.
- **msg:"MISTIC malware trojan detected"**. El "msg" configura el engine de logs y de alertas para escribir el mensaje indicado junto al paquete que cumple la regla especificada. En este campo podemos definir concretamente qué mensaje.
- **sid:124444**. El ID de la regla.
- **rev:1**. El número de revisión de la regla: un control de versión para su correcto mantenimiento.
- **priority:4**. El "priority" indica mediante un número entero la prioridad de la regla. El rango de prioridades va desde 1 (alto) hasta 4 (muy bajo).
- **classtype:trojan-activity**. El "classtype" se utiliza para categorizar una regla al detectar un ataque. De esta forma, se puede determinar que el ataque es parte de una clase de ataques. En este caso, la regla lo cataloga como un ataque relacionado con la actividad de un troyano.
- **flow:to_server,established**. El campo "flow" permite a la regla aplicarse solo en ciertas direcciones del flujo de tráfico. En este caso, tenemos que la regla se aplica cuando haya peiticiones como cliente en conexiones TCP establecidas.
- **content:"User-Agent";http_header**. El campo "content" permite al usuario configurar las reglas para buscar un contenido especifico en el *payload* del paquete. De esta forma podemos buscar contenido específico en las tramas a analizar. "content" permite una serie de palabras clave que matizan la naturaleza del contenido, en este caso, se restringe la búsqueda de la búsqueda a las cabeceras HTTP.

- **pcre**. El campo “pcre” permite usar expresiones regulares.

```
"/^User-Agent\s*:[^\n]*malw4r3/"
```

  Esta expresión regular permite buscar en la cabecera “User-Agent” del protocolo HTTP el valor que contenga la palabra “malw4r3”, puesto que es una característica común de todas las pcaps del *downloader* en la petición de descarga del archivo “update.zip”.

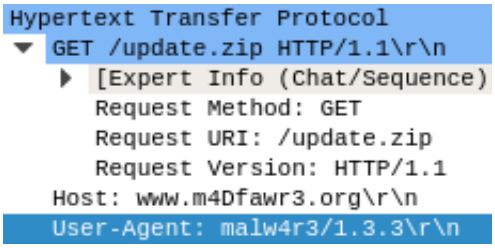



Figura 6: Cabecera “User-Agent” de la trama GET

```
root@mistic:~# snort --pcap-dir=pcaps/ -A console -q -c /etc/snort/snort.conf
10/20-22:29:16.746653  [**] [1:124444:1] MISTIC malware trojan detected [**] [Cl
assification: A Network Trojan was detected] [Priority: 4] {TCP} 10.0.0.1:45474
-> 58.14.0.1:80
10/20-22:29:53.717257  [**] [1:124444:1] MISTIC malware trojan detected [**] [Cl
assification: A Network Trojan was detected] [Priority: 4] {TCP} 10.0.0.1:59208
-> 61.29.128.91:80
10/20-22:30:21.291925  [**] [1:124444:1] MISTIC malware trojan detected [**] [Cl
assification: A Network Trojan was detected] [Priority: 4] {TCP} 10.0.0.1:43794
-> 123.64.99.200:80
10/20-22:30:41.400607  [**] [1:124444:1] MISTIC malware trojan detected [**] [Cl
assification: A Network Trojan was detected] [Priority: 4] {TCP} 10.0.0.1:33925
-> 166.111.111.166:80
10/20-22:30:56.707807  [**] [1:124444:1] MISTIC malware trojan detected [**] [Cl
assification: A Network Trojan was detected] [Priority: 4] {TCP} 10.0.0.1:37782
-> 203.100.192.58:80
root@mistic:~#
```

Figura 7: Ejecución de Snort con la nueva regla añadida sobre el directorio pcap/

# 3.

## 3.1.

Tal como se puede apreciar en la figura 7 no hay ninguna detección relativa a la sexta pcap cuya IP de destino es 210.14.128.7.

## 3.2.

La muestra número 6 ha conseguido evadir la regla de Snort. De alguna forma, ha evitado que la expresión regular fuera evaluada correctamente en el proceso de detección y por eso no ha activado la alarma.

## 3.3.

Snort no parece mostrar ningún tipo de alerta sospechosa.

## 3.4.

La técnica utilizada por el *downloader* se denominda*Fragmentation reassembly timeout attack* que, tal como su nombre indica, consiste en utilizar la fragmentación de paquetes en el protocolo IP.
Dentro del protocolo IP existe la posibilidad de fragmentar el paquete y mandarlo en trozos más pequeños que son reordenados y reconstruidos en el destinatario. Este ataque se puede utilizar de tal forma que se puede eludir las reglas de Snort y el destinatario reordena la trama final ejecutando el ataque. Como se puede apreciar, el ataque se divide en distintos fragmentos que son

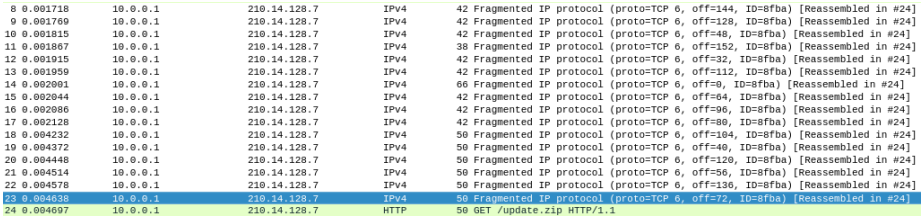



Figura 8: Fragmentación de paquetes IP reordenados en la trama 24

reconstruidos al final. Esta reconstrucción pasa desapercibida para el NIDS y se puede efectuar la descarga del archivo malicioso.

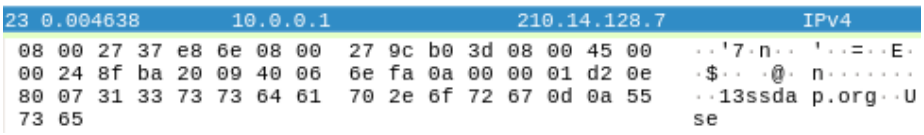



Figura 9: Contenido de la trama 23

Se puede observar que los datos que utilizamos para determinar el contenido malicioso (User-Agent: malw4r3) queda dividido en 2 tramas diferentes (fig. 9, fig. 10) y la expresión regular para detectarlo no tiene efecto.

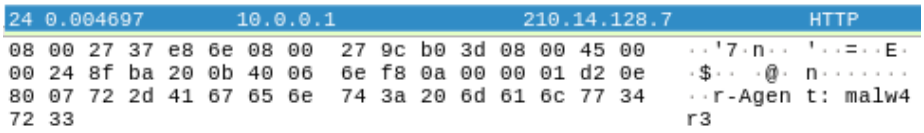



Figura 10: Contenido de la trama 24

### 3.5.

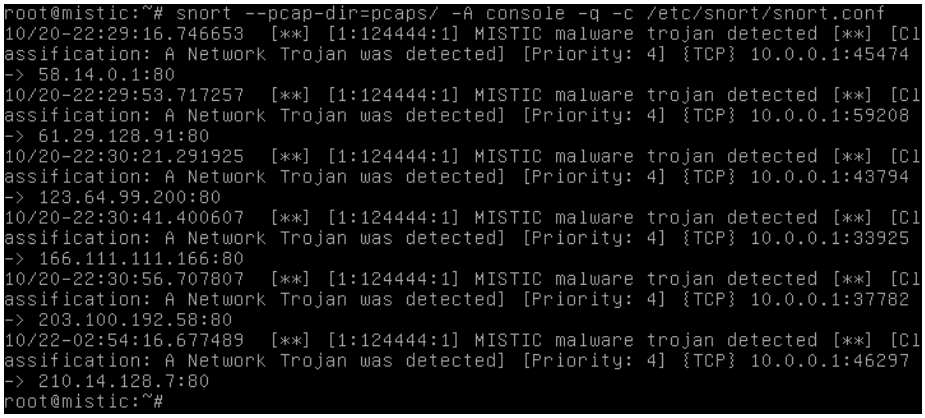



Figura 11: Ejecución de Snort con la modificación del preprocesador

Al cambiar la política del preprocesador, se cambia la manera en la que se reordenan los fragmentos IP en el NIDS. La política actual reordena las tramas de manera que la expresión regular surge efecto y podemos detecta el texto que nos indica que el paquete es, efectivamente, un malware *downloader*.

### 3.6.

Los preprocesadores permiten ampliar la funcionalidad de Snort de manera modular. Concretamente, los preprocesadores son programas que actúan antes que el motor de detección y después de que el paquete se haya decodificado. De esta forma, el paquete puede ser analizado de manera independiente. El preprocesador *frag3* es un módulo de desfragmentación de IP basado en objetivos (*target-based*).

### 3.7.

Por defecto, Snort tiene la política de fragmentación First, correspondiente a la plataforma Windows. Esta política reordena los paquetes por orden de llegada estricto y esta política no podía detectar el ataque. La política de Linux, en cambio, reordena los paquetes de forma diferente, una que sí permite detectar el malware.

## 3.8.

En Snort hay 2 directivas del preprocesador *frag3*, la global y la configuración del motor. En la última, el usuario puede estipular qué política de desfragmentación se tiene que configurar para una IP de destino.
El preporcesador de *frag3* es un módulo de desfragmentación para Snort basado en IP objetivo. En un sistema basado en objetivos se modelan los objetivos reales en la red en vez de modelar los protocolos y buscar ataques en ellos. Los ataques IPs se diseñan para distintos sistemas operativos, de esta forma, se puede modelar el sistema objetivo en una situación más realista.

# 4.

Esta regla detecta una vulnerabilidad en la función "use_syslog()" en el sistema de impresión LPRng v3.6.24 que permite al atacante ejecutar comandos arbitrarios.

- **alert**. Generación de una acción de tipo "alert".
- **tcp**. Tipo de paquete sobre el que debe actuar.
- **$EXTERNAL_NET any → $HOME_NET 515**. Dirección del paquete. En este caso cualquier paquete que venga desde el exterior por cualquier puerto hacia la red interna con el puerto 515 como destino.
- **msg:"EXPLOIT Redhat 7.0 lprd overflow"**. Mensaje que generará la alerta en caso de cumplirse la regla.
- **flow:to_server,established**. Aplica la regla en conexiones TCP establecidas como cliente.
- **content:"XXX %.172u %300|24|n"**. Busca en la trama este contenido.
- **reference:bugtraq,1712**. El campo "reference" permiten a la regla incluir referencias a sistemas de identificación de ataques externos. En este caso tenemos una referencia al sistema bugtraq.
- **reference:cve,2000-0917**. Esta es otra referencia, en este caso al Common Vulnerabilities and Exposures.
- **classtype:attempted-admin**. La categoría de este ataque pertenece a la clase de intentar ganar privilegios de administrador.
- **sid:302**. ID de la regla.
- **rev:9**. Número de revisión.

La regla busca cualquier trama que venga desde la red externa al puerto 515 cuyo contenido sea: "XXX %.172u %300|24|n". De esta forma se protege al sistema de que alguien pueda explotar esta vulnerabilidad.

# 5.

## 5.1.

En términos generales el script inicializa una cadena llamada WHITELIST donde se recogen algunas reglas conrcetas. Se inicializan las cadenas por defecto (INPUT, OUTPUT, FORWARD) para que sigan una política de rechazo por defecto (DROP).

A continuación se definen una serie de normas para la cadena INPUT que acepta conexiones SSH y rechaza aquellas conexiones que consistan en más de 6 peticiones en 30 segundos. También añade a la WHITELIST una serie de IPs de origen y rechaza todo lo demás en lo relativo a esta cadena.

Finalmente añade a la cadena de OUTPUT los paquetes relativos a conexiones en curso.

## 5.2.

```
#/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
```

```
iptables -F
iptables -X
```

Figura 12: Vacía (*flush*) todas las cadenas de la tabla y elimina todas las cadenas que no sean por defecto.

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

Figura 13: Configura la política para las cadenas por defecto. En este caso la política es rechazar todo por defecto.

```
iptables -N WHITELIST
```

Figura 14: Crea una cadena nueva llamada “WHITELIST”

A continuación todos los comandos hacen referencia a la acción de añadir reglas a una u otra cadena (-A).

En esta sección, concretamente, hablamos siempre de añadir reglas a la cadena por defecto de INPUT. Esta cadena hace referencia a todo el tráfico de entrada.

```
iptables -A INPUT -i lo -j ACCEPT
```

Figura 15: Se acepta tráfico que provenga de la interface de loopback.

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Figura 16: Se aceptan paquetes que pertenezcan a conexiones ya existentes y a paquetes que estén relacionados a las mismas como los errores ICMP. El flag **-m** en iptables sirve para cargar extensiones que ayudan a determinar la naturaleza del paquete. El flag junto con la extensión hacen una comparación con el paquete. Por ejemplo, en la siguiente regla tenemos el state **NEW** que lo que hará será permitir evaluar el paquete con la extensión **NEW** que nos dice si el paraquete crea una nueva conexión.

Recordemos que las reglas se evalúan de arriba abajo. Empezamos con las reglas de creación de WHITELIST y configuración de políticas por defecto. Luego se añaden las reglas que permiten aceptar paquetes de manera siendo cada vez más concretas restrictivas. Primero se especifican las reglas de la cadena INPUT que tienen relación con el protocolo SSH y conexiones previamente establecidas, después se añaden a la lista blanca los paquetes cuyo origen sean unas IPS concretas y se rechazan los demás. Finalmente, a la lista de salidas las conexiones previamente establecidas.

```
iptables -A INPUT -p tcp --dport ssh -j WHITELIST
```

Figura 17: Todo paquete del protocolo TCP cuyo destino sea el puerto SSH se redirige a la cadena WHITELIST

```
iptables -A INPUT -p tcp --dport ssh -m state --state NEW -m recent
     --set
```

Figura 18: Todo paquete del protocolo TCP cuyo destino sea el puerto SSH y que el paquete cree una nueva conexión. Además, busca en la cadena la IP del paquete y la actualiza o, si no la encuentra, la añade.

```
iptables -A INPUT -m recent --update --seconds 30 --hitcount 6 -j
    LOG
```

Figura 19: El estado **recent** permite crear una lista dinámica de IPs y luego hacer búsquedas en ella de distintas formas. El flag **update** mira si el paquete se encuentra actualmente en la lista y, si lo está, actualiza su timestamp. El flag **seconds** delimita la búsqueda a paquetes de la lista que hayan sido actualizados en la franja de segundos indicada. El flag **hitcount** delimita aún más la búsqueda a paquetes que hayan recibido un valor mayor o igual a este. **-j LOG** salta a la acción de LOG. Los paquetes que cumplan estas condiciones quedan logueados en el sistema de logs.

```
iptables -A INPUT -m recent --update --seconds 30 --hitcount 6 -j
    DROP
```

Figura 20: Es la misma instrucción que antes, sólo que esta vez la acción es de **DROP**. Es decir, los paquetes quedan primero logueados e inmediatamente después se descartan

```
iptables -A INPUT -p tcp --dport ssh -m state --state NEW -j ACCEPT
```

Figura 21: Paquetes del protocolo TCP cuyo destino sean el puerto SSH y que inicen una nueva conexión serán aceptados.

```
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
```

Figura 22: Mete en la WHITELIST la regla que permite que los paquetes cuyo origen sean 10.0.0.0/16, 10.1.0.0/16, 10.2.0.0/16, 10.3.0.0/16 sean redirigidos a la siguiente regla de la cadena que ha sido llamada previamente a esta

```
iptables -A WHITELIST -j DROP
```

Figura 23: Si se llega a esta regla en WHITELIST, quiere decir que se rechazan los paquetes.

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

Figura 24: Añade a la cadena de salida la regla que acepta aquellos paquetes que inicien una conexión relacionada con una conexión existente o relacionados con una conexión donde se hayan visto paquetes en ambas direcciones.

## 5.3.

El comando que permitiría registrar conexiones SSH que no estén dentro de la lista blanca sería

```
iptables -A WHITELIST -p tcp --dport ssh -j LOG
```

y se tendría que situar antes de la regla que rechaza los paquetes que no cumplan algunas de las condiciones anteriores:

```
iptables -A WHITELIST -j DROP
```