Trabajo de Promoción - SYPER

## Ivan Karl, 12160/3

## Ruta Santiago, 12003/0

## PARTE (A) - Tarea:

**1. Configurar adecuadamente los servicios que se ofrecen hacia Internet, de modo que no revelen**

**más información que la necesaria.**

En base a los conocimientos adquiridos en la Entrega 1, se detectó que el host **User6** [192.168.6.21] de la sucursal tenía el servicio de DNS activo y que en la casa matriz tanto el **resolverDNS** [192.168.7.14] como el servidor autoritativo de la zona “syper.edu”, **ns-syper-edu** [192.168.7.34], respondían a consultas recursivas realizadas por hosts externos a la organización y mostraban la versión del **bind**.

Para solucionar el primer issue se desactivo el servicio **bind** en el User6 mediante el comando **service bind9 stop**, mientras que para el segundo se reemplazo el archivo de configuración del **bind**, named.conf.options, en **resolverDNS** y **ns-syper-edu** por uno con el siguiente contenido:

acl "trusted" {

193.81.6.0/24;

193.81.7.0/28;

193.81.7.16/28;

193.81.7.32/28;

};

options {

directory "/var/cache/bind";

listen-on-v6 { none; };

alow-query { trusted; }; //solo pueden relizar consultas los hosts de 'trusted'

version "NO-TE-DIGO"; //'nueva versión' de bind

alow-query-cache { trusted };

recursion: yes; //permite recursión

alow-recursion { trusted; }; //pero solo a los hosts de 'trusted'

};

Previamente al reemplazo del archivo se apagó el servicio **bind** mediante el comando **service bind9 stop** y posteriormente se reanudó con **service bind9 start** para evitar inconvenientes con los procesos en ejecucion.

**2. Configurar reglas de firewall en donde crea necesario de modo que, desde Internet, sólo se puedan acceder a los servicios públicos de la organización.**

**3. Configurar reglas de firewall en donde crea necesario de modo que desde las distintas redes de la organización solo se pueda acceder a lo necesario para que:**

**a. Los usuarios puedan acceder a los servicios internos.**

**b. Los servicios internos funcionen y estén disponibles para ser accedidos desde la red de la organización (casa matriz y sucursal).**

**c. Los usuarios puedan acceder a Internet.**

Al no tener la topología ningún firewall configurado, tanto en la casa matriz como en la sucursal se eligieron los routers de frontera de cada una, el **n28** y el **n18** respectivamente, para implementarlos. En el **n28** se determinó una política de forwarding restrictiva por defecto, en la cual se rechaza toda conexión que no esté explícitamente permitida. En la lista blanca se agregaron políticas para permitir todas las conexiones salientes de los hosts de la red interna, conexiones entrantes de la sucursal y conexiones entrantes provenientes de cualquier host hacia los servicios públicos de la organización. Los servicios públicos presentes son: la página institucional, servida en el puerto 80/tcp de **n35** [193.81.7.36], el servicio de DNS en el puerto 53/udp y 53/tcp de **n32** [193.81.7.34] y el servicio de email presente en los puertos 25/tcp y 993/tcp de **n34** [193.81.7.35]. Finalmente se permiten el ingreso de paquetes el exterior pertenecientes a conexiones establecidas o relacionadas.

Comandos ejecutados en **n28**:

iptables -P FORWARD DROP

iptables -A FORWARD -i eth0 -j ACCEPT

iptables -A FORWARD -s 193.81.6.0/24 -d 193.81.7.0/26 -j ACCEPT

iptables -A FORWARD -d 193.81.7.36 -p tcp --dport 80 -j ACCEPT

iptables -A FORWARD -d 193.81.7.34 -p udp --dport 53 -j ACCEPT

iptables -A FORWARD -d 193.81.7.34 -p tcp --dport 53 -j ACCEPT

iptables -A FORWARD -d 193.81.7.35 -p tcp -m multiport --dports 25,993 -j ACCEPT

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

En el **n18** también se determinó una política de forwarding restrictiva por defecto, así como también se permite la navegación hacia el exterior desde los hosts presentes en la red de la sucursal y el retorno de paquetes con condición de establecido o relacionado. También se agregó una regla que permite que los hosts presentes en la casa matriz accedan a los hosts de la sucursal.

Comandos ejecutados en **n18**:

iptables -P FORWARD DROP

iptables -A FORWARD -i eth0 -j ACCEPT

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

iptables -A FORWARD -s 193.81.7.0/26 -d 193.81.6.0/24 -j ACCEPT

**4. Configurar una VPN para proteger el tráfico intercambiado entre la casa matriz y la sucursal (copia**

**de claves, configuraciones y reinicio del servicio).**

Para establecer una red virtual privada (VPN) entre la casa matriz y la sucursal se configuró el programa **Ipsec** en los routers de frontera de ambos, **n28** [ext (eth1): 193.81.11.1 | int (eth0): 193.81.7.50] y **n18** [ext (eth0): 193.81.17.1, int (eth1): 193.81.6.1] respectivamente, con el script autogenerado en la herramienta **core** y los siguientes parámetros:

En **n18**:

keydir="/etc/core/keys"

certname=ipsec1

tunnelhosts="193.81.17.1AND193.81.11.1"

T1="193.81.6.0/24AND193.81.7.0/28 193.81.6.0/24AND193.81.7.16/28 193.81.6.0/24AND193.81.7.32/28"

En **n28**:

keydir="/etc/core/keys"

certname=ipsec1

tunnelhosts="193.81.11.1AND193.81.17.1"

T1="193.81.7.0/28AND193.81.6.0/24 193.81.7.16/28AND193.81.6.0/24 193.81.7.32/28AND193.81.6.0/24"

La especificación anterior determina que todo el tráfico entre 193.81.17.1 y 193.81.11.1 se encriptará, y se desencriptará una vez que pase a las redes internas de cada routes. El script copia las claves y certificados generadas anteriormente a la carpeta /etc/core/keys para que una vez inicializado el servicio estas puedan ser intercambiadas.

NOTA: configurando Ipsec gráficamente se pudo crear la vpn, pero no se pudo lograr el mismo resultado utilizando el script de configuración, por lo que el archivo .imn ya tiene la vpn creada desde su inicio. Igualmente se deja el script utilizado con la invocación a esa función comentada para posterior revisión.

**5. Configurar un Web Application Firewall (modSecurity) en los servidores web de la organización de modo que, cuando se detecte un ataque, se muestre una advertencia al usuario.**

modSecurity es un firewall de aplicaciones web que se ejecuta como módulo del servidor web Apache y que se distribuye bajo la licencia GNU. Su función principal es detectar y prevenir intrusiones en aplicaciones web.

Los servidores presentes en la topología que poseen páginas web activas son **www-syper-edu** [193.81.7.36] e **intranet-syper-edu** [193.81.7.12], por lo que el script de inicialización instala el paquete **libapache2-modsecurity** en la máquina virtual para que sea accesible a estos y luego se habilita con el comando **a2enmod security2**. Luego realiza las configuraciones, utilizando las reglas presentes en <https://github.com/SpiderLabs/owasp-modsecurity-crs>, y movimientos de archivos pertinentes, entre ellos la página de error custom que se muestra al detectar acciones sospechosas en los sitios. Finalmente se resetea el servicio apache2 en cada uno de los hosts para que entren en vigencia los cambios.

Para el testeo de esta mejora se verificó que, una vez activado el módulo, el servidor redirige al usuario a la página de error en cuanto este intenta efectuar un ataque del tipo XSS en cualquiera de los dos servidores.

**6. Implementar las mejoras de seguridad propuestas en las distintas entregas previas (servidores, DNS de syper / apagar servidores que no deberían estar encendidos, etc).**

Punto implementado durante la realización de los anteriores.

**7. Configurar un IDS de red en la interfaz interna del router que conecta a Internet las redes de la organización para detectar ataques y scannings originados desde dentro de la organización.**

Para llevar a cabo este punto se eligió utilizar **Snort** como Network Intrusion Detection System ya que fue visto durante el dictado de la cátedra. Para configurar la herramienta se utilizan dos parámetros principales, HOME\_NET, el cual indica cuál es la red que se quiere proteger, y EXTERNAL\_NET, el cual indica cuál es la red que tiene que monitorear buscando comportamientos sospechosos.

Se decidió correr el daemon de **snort** en los routers de frontera de la sucursal y de la casa matriz. Para **snort** comience a ejecutarse lo que se hizo fue copiar un script en cada router que contiene los parámetros de ejecución del programa y luego se ejecutó localmente en el nodo.

En **n18**:

NETS='193.81.6.0/24'  
 snort -D \  
 -S HOME\_NET="!${NETS}" \  
 -S EXTERNAL\_NET="${NETS}" \  
 -c /etc/snort/snort.conf \  
 -i eth0 \  
 -l var.log/snort \  
 -K ascii

En **n28**:

NETS='[193.81.7.0/28,193.81.7.16/28,193.81.7.32/28,193.81.7.48/28]'   
snort -D \  
 -S HOME\_NET="!${NETS}" \  
 -S EXTERNAL\_NET="${NETS}" \  
 -c /etc/snort/snort.conf \  
 -i eth0 \  
 -l var.log/snort \  
 -K ascii

Como se puede ver, el parámetro HOME\_NET se configuró para ambos routers con la red externa, y se seleccionaron como redes no confiables las internas a cada uno, para que estas sean las que se monitoreen.

## PARTE (B) - Testing:

Se agrego un script `test\_runner.sh` el cual ejecuta todos los tests provistos.

Para ejecutar los tests independientemente debe uno pararse en la carpeta contenedora y ejecutarlos directamente, esto se debe a que agregamos un archivo de ‘helpers’ (funciones que cambian el color del texto) para facilitar la lectura de los tests en la consola:

cd tests

./test\_dns

./test\_firewall

./test\_ids

./test\_waff

Puntos 1, 2 y 3 son corroborados por los tests de dns y vpn.

Punto 4 es corroborado por `test\_vpn`.

Punto 5 es corroborado por `test\_ids`.

Punto 6 es corroborado por `test\_waf`.