

PORT - KNOCKING

Jonathan Barajas
jonathan [DOT] barajas [AT] indiseq
[DOT] net



<http://www.indiseq.net>

1 - INTRODUCCIÓN

2 - ¿QUÉ ES PORT-KNOCKING?

3 - PORT-KNOCKING CASERO (TCPDUMP + SENDIP)

4 - PROBLEMAS INTRÍNSECOS AL PORT-KNOCKING

5 – IMPLEMENTACIONES

6 – USOS REALES

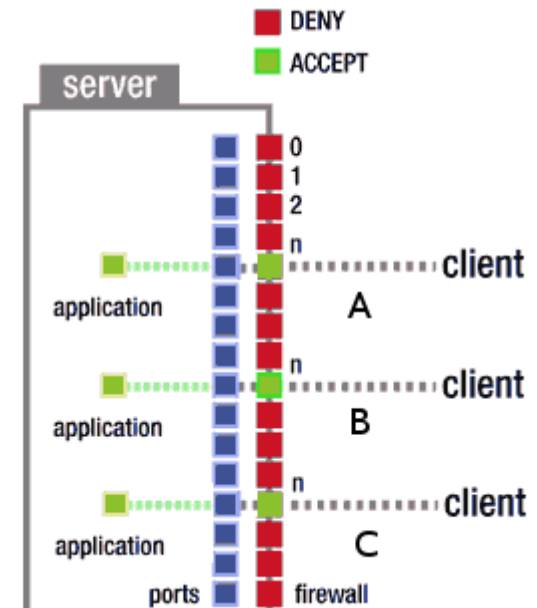
7 - REFERENCIAS

Introducción

En la parte servidor las aplicaciones escuchan en un (o varios) puerto a la espera de las peticiones de los clientes.

Los clientes envían intentos de conexión al servidor y a un determinado puerto, donde les espera el servicio.

Cuando un paquete llega al servidor, el cortafuegos consulta los filtros que tiene predefinidos, y comprueba que el paquete se ajuste a ellos.



Problemas de los sistemas actuales

Los cortafuegos de hoy en día son capaces de realizar infinidad de filtros:

- Por dirección IP
- Por dirección MAC
- Por protocolo
- Por puertos
- etc

Pero se encuentra limitados a la hora de realizar filtros a mayor nivel:

- ¿Quién se conecta?
- ¿Desde donde se conecta?
- ¿Direccionamiento dinámico?

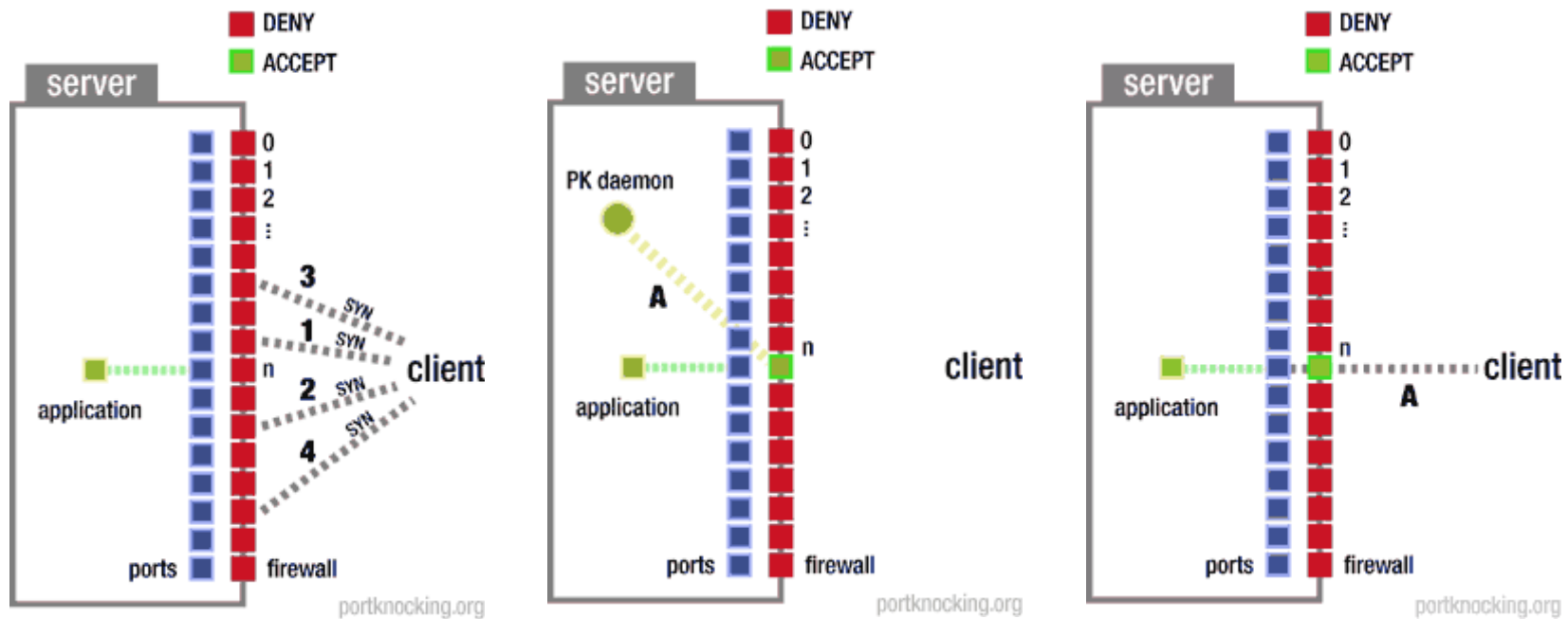
Y una vez que el cortafuegos no ha podido impedir una conexión al servicio, surgen otros problemas:

- Ataques de adivinación de contraseñas
- Exploits
- Fugas de información
- etc.

Posible solución: **Port
Knocking**

¿Qué es portknocking?

PortKnocking es un método que permite mediante el envío de una serie de paquetes predefinidos a puertos cerrados, reajustar las reglas de un firewall para permitir determinadas conexiones.



¿Qué aporta el Port-Knocking?

- Una capa de seguridad extra:
 - Filtrado de usuarios
 - Integración con el cortafuegos
 - Oscurantismo
- Protección de los servicios, permitiendo la conexión solo en el momento en el que se vayan a utilizar
- ¿ Ejecución remota de comandos ? ¿Administración silenciosa?

Portknocking casero (Sendip y tcpdump)

Objetivo:

Implementar un Portknocking que escuche paquetes TCP SYN :

- Que vayan dirigidos al rango de puertos comprendido entre el 1000 y el 10999
- Con número de secuencia 17

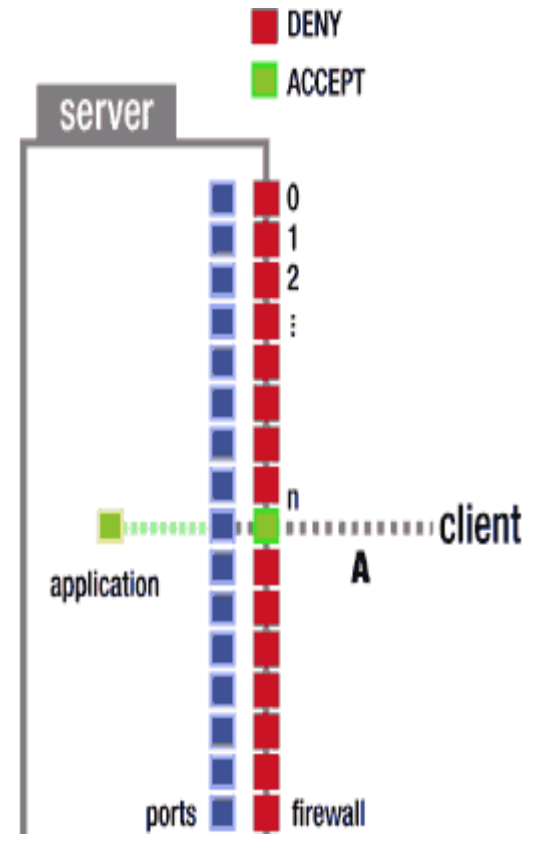
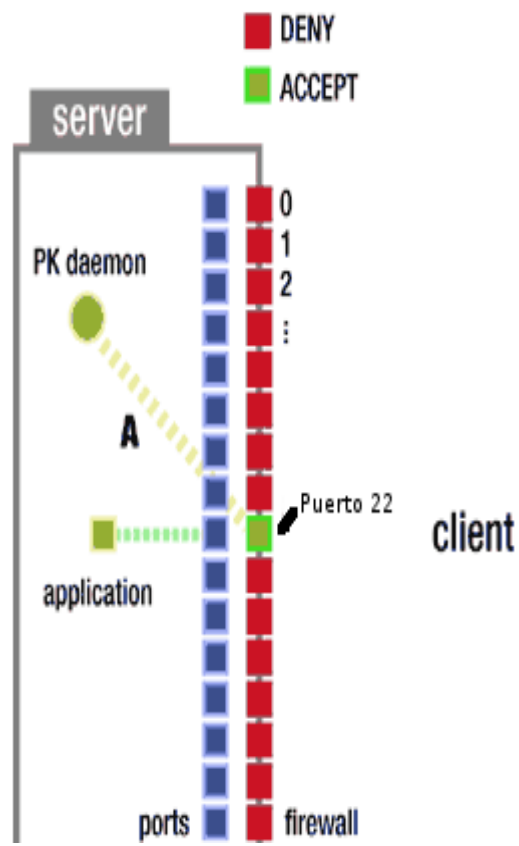
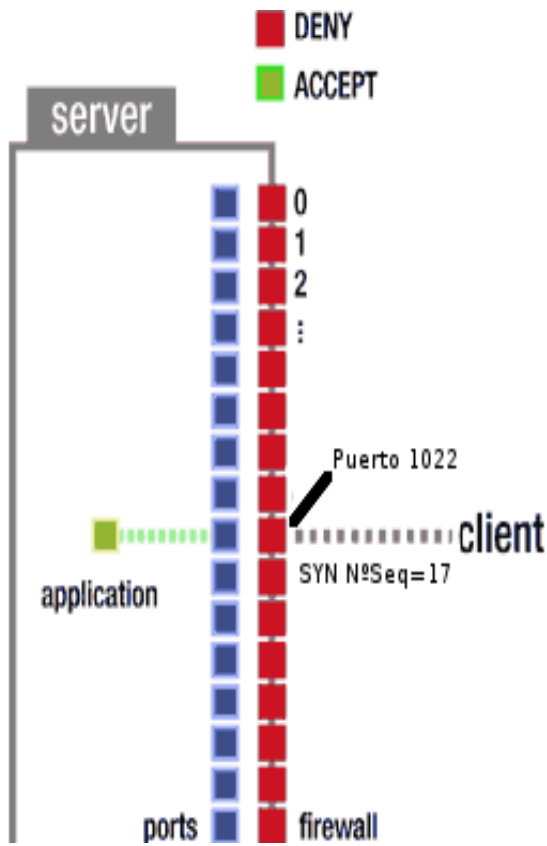
Resultado:

- Se abrirá el puerto que el paquete SYN indique como su puerto destino sin incluir el prefijo 10.

Ej: Paquete SYN al puerto 1022 – Abrirá el puerto 22

- El puerto permanecerá abierto durante 10 segundos a la dirección de origen del paquete SYN

Ejemplo:



1 - Configuración del firewall:

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

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED
-j ACCEPT
iptables -A INPUT -s 0/0 -d 0/0 -p udp -j DROP
iptables -A INPUT -s 0/0 -d 0/0 -p tcp --syn -j DROP
```

3 - Recepción y filtrado del paquete

```
tcpdump -v -O "tcp[2:2] >= 1000 and tcp[2:2]
<= 10999 and tcp[4:4] = 17 and tcp[tcpflags] &
tcp-syn != 0"
```

tcp[2:2] – Un campo de 2 bytes en la cabecera TCP a partir del byte número 2 (tcp [principio:longitud])

5 - Uniendo las piezas

```
tcpdump -l -O "tcp[2:2] >= 1000 and tcp[2:2] <= 10999 and tcp[4:4] = 17 and tcp[tcpflags] &
tcp-syn != 0" | sed -u 's/. *IP \([0-9]*\.[0-9]*\.[0-9]*\.[0-9]*\) .* 10*\([0-9]*\) : S.*\1 \2/' | xargs -t -l
-j bash -c "guard.sh {}" &
```

Paquete capturado por tcpdump:

```
11:30:57.016986 IP 192.168.1.210.1000 > 192.168.1.203.1022: S 17:17(0) win 65535
```

2 - Creación del knock

```
sendip -p ipv4 -p tcp -is 192.168.1.210 -ts 1000 -td 1022 -tfs 1 -tn 17
192.168.1.203
```

-is : Dirección IP origen
-ts : Puerto origen
-td : Puerto destino
-tfs : Flag SYN
-tn : Número de secuencia

4 - Reajuste de las reglas de firewall ante un paquete válido

```
# guard.sh
# abrimos durante 10 segundos
/sbin/iptables -I INPUT -j ACCEPT -i eth0 -p tcp -s $1
-dport $2
sleep 10
# cerramos
/sbin/iptables -D INPUT -j ACCEPT -i eth0 -p tcp -s $1
```

```
iptables -I INPUT -j ACCEPT -i eth0 -p tcp -s $1
-dport $2
```

Problemas intrínsecos a Portknocking

- **Spoofing**: Envío de paquetes correctos, con direcciones origen falsas, permitiendo así el acceso desde otras direcciones IP.

Posible solución: Implementaciones como Doorman permite asignar un rango de direcciones IP a cada usuario (grupo) desde donde se pueden conectarse.

- **Ataques de reproducción (replay)**: Este ataque se produce cuando esnifando una comunicación se captura un knock válido, y se reenvía (el mismo o retocado) con el objetivo de lograr el acceso.

Posibles soluciones: - Generación de knockers cifrados con elementos dinámicos.

- Cortinas de humo (Ocultar las secuencias entre otros paquetes)

Problemas asociados a la autenticación: El hecho de que el servidor no pueda enviar ningún paquete hasta que el cliente se identifique, dificulta la posibilidad de uso de una autenticación más fuerte.

Posible solución: Uso de contraseñas validas para un acceso.

Doorman ([http://doorman.sourceforge.net /](http://doorman.sourceforge.net/))

Implementación creada por Martin Krzywinski's, que propone el uso de un solo paquete UDP.

Los paquetes UDP contienen un hash MD5 que está formado por



Características:

Doorman identifica diferentes permisos en base a usuarios y grupos.

Medidas contra ataques de reproducción. Número aleatorio valido para una vez

Compatibilidad con gran número de cortafuegos (ipchains, iptables, pf)

Versión del knocker para windows, FreeBSD, OpenBSD, NetBSD, MacOS X

Cerberus (<http://silverstr.ufies.org/blog/archives/000625.html>

)

Objetivos:

- Evitar el uso de clientes específicos (Uso de comandos estandar)
- Métodos de autenticación
- Uso de paquetes clave más difíciles de identificar que los TCP

(ICMP)

| UserID | Hash | |
|------------|------------------|------------|
| dead 42 01 | f0b70bc031a365e9 | ccf47bea |
| Iniciador | AcciónID | DirIP(Hex) |

Características:

- Utiliza paquetes ICMP estandar
- No necesita de clientes especiales (ping, md5sum, date)

```
date +%d%m%y%k%Msome_seedmy_pincode204.244.123.234 | md5sum | cut -c 17-32
```

- No sobrecarga mucho el sistema, ya que no busca patrones específicos en el sniffing o en los logs.
- Hash MD5 (fecha y hora, identificativo servidor, contraseña y ip permitida)
- Uso de “one time password” para evitar ataques de reproducción
- En el paquete se incluye un ID de acción definido en el servidor.

webknocking (<http://www.webknocking.de/>)

Motivos de su desarrollo:

- Posibilidad de obtener la secuencia de knocks esnifando la red
- La necesidad de disponer de un determinado cliente
- Tener un servicio dedicado a la espera de peticiones

Definición:

Webknocking es una implementación de portknocking que en vez de utilizar puertos, utiliza una página web. El único requisito es tener un servidor Web. Así nace el concepto de webknockingsequence. El servidor web, está configurado para logar todas las peticiones, y cuando se produzca la correcta, se abrirá un puerto en el servidor.

Para los posibles ataques de reproducción se implementa:

- Secuencias dinámicas (difícil de acordarse)ç
- Secuencias + reto pregunta respuesta
- Https

Otros implementaciones

pasmal (<http://www.sourceforge.net/projects/pasmal/>) : Puede utilizar paquetes TCP/UDP/ICMP a puertos abiertos o cerrados indistintamente. Dispone de una página web de administración. Crear “pantallas de humo” (camufla los knocks entre gran cantidad de paquetes aleatorios) para dificultar ataques.

fwknop (<http://www.cipherdyne.org/fwknop/>) : Añade al tradicional sistema, un detector de sistema operativo (p0f – Passive OS Fingerprinting tool). Se pueden utilizar paquetes TCP/UDP/ICMP y en vez de esnifar el tráfico, parsea los logs de los cortafuegos. También es capaz de distinguir entre apertura de puertos y comandos.

winportknocking (<http://sourceforge.net/projects/winportknocking>) : Portknocking para windows.

wknock (<http://www.rstack.org/oudot/wknock/>) : Permite ocultar puntos de acceso wireless. Existe una versión para Linux y también para n para OpenWRT (Distribución Linux para Routers Wireless)

Usos reales

El Port Knocking puede ser muy útil en los siguiente ambientes:

- **Dispositivos con servicios de administración abiertos.** Estos servicios se suelen utilizar poco, y la mayor parte del tiempo se encuentran expuesto a terceros sin necesidad. Portknocking permite abrir el servicio solo en el momento que sea necesario.

- **Servidores de uso privado.** Para un número no muy elevado de usuarios, es posible utilizar estas técnicas para ocultar el servicio a terceros, y para evitar los ataques más recientes antes de que se produzca el parcheado.

- **Servicios sobre los que no se realizan revisiones de seguridad continuas.** Portknocking permite salvaguardar la seguridad de aquellos servicios que son vulnerables a ataques.

- **Servicios públicos.** Debido a la necesidad de conocer la secuencia de entrada para el acceso

- **Servicios a gran número de usuarios:** Por el esfuerzo que requeriría la actualización y la gestión de los usuarios.

Referencias

<http://www.portknocking.org> : Página de referencia obligada en temas de Portknocking

http://en.wikipedia.org/wiki/Port_knocking : Wikipedia

<http://en.hakin9.org/> : Krzywinski M (2005) Port Knocking From the Inside Out. hakin9

<http://netsecurity.about.com/cs/generalsecurity/a/aa032004.htm> : Usos buenos y “malos” del portknocking