

# Covert Channels

26 de octubre de 2010 Por [Joaquín Moreno](#)

El encubrimiento de canales o *Covert Channels* es una técnica de evasión que permite a un atacante enviar información empleando para ello las cabeceras de los protocolos de comunicación. En esta entrada trataremos el encubrimiento de canales en el protocolo TCP/IP, y proporcionaremos una herramienta, **CovertShell**, diseñada para la prueba de concepto. Pueden localizar los fuentes al final de la entrada.

El protocolo TCP/IP presenta una serie de cabeceras que normalmente son inicializadas por el cliente para mantener o numerar una comunicación. La técnica *covert channels* aprovecha dichos campos para asignarle valores de tal forma que la máquina destino interprete dichos campos no como una forma de mantener una comunicación o aportar información sobre ésta, sino para obtener datos.

Un ejemplo interesante fue desarrollado por Craig H. Rowland en su *paper* de 1996: *Covert Channels in the TCP/IP protocol Suite*, donde éste creaba una pequeña herramienta cliente/servidor "CovertTCP" de no mas de 500 líneas que permitía la transferencia de ficheros entre cliente/servidor empleando para ello únicamente los campos SEQ, ACK del protocolo TCP y el campo ID del protocolo IP. De esta forma la información iba en las cabeceras de los protocolos y no en el contenido del paquete o *payload*.

A raíz de esta idea, siempre se ha pensado en la posibilidad de ir más allá y usar esta técnica para el envío de ordenes de comando. Algo que podría ser usado tanto para puertas traseras (*backdoors*) como para redes *botnet*. Imagínense que tenemos un servidor que ha sido comprometido al cual se le ha instalado un servicio que espera paquetes en un cierto puerto con ciertos valores en su cabecera. Cuando le llega un paquete que cumple esos requisitos, procesa los valores de las cabeceras convirtiendo dichos valores en ordenes de comando, que son posteriormente ejecutadas.

Siguiendo este modelo hemos creado una pequeña herramienta para la prueba de concepto a raíz del código fuente del CovertTCP, denominada CovertShell, donde disponemos de un servicio que emplea un *socket RAW* de tal forma que al recibir en el puerto indicado paquetes con el bit ACK y sin el bit SYN, obtiene el identificador de la cabecera IP, el cual es el número identificativo correspondiente a una letra en **código ASCII**. Dicha letra es almacenada en un vector. Cuando llegan 3 paquetes cuyo ID es 255, el servidor ejecuta la orden de comando que hay almacenada en el vector. Por tanto disponemos de un servidor que no abre ningún puerto, que realmente escucha el tráfico y que es capaz de interpretar la cabecera IP para generar comandos. Dicho de otra forma, tenemos una puerta trasera empleando *covert channels* para comunicarse. ¿Lioso? Veamos un ejemplo de funcionamiento:

Compilamos y ejecutamos el servicio que espere paquetes en el puerto 8888 en la máquina víctima (servidor):

```
# gcc servidor.c -o servidor
# ./servidor -port 8888
```

En otra consola comprobamos que el proceso está levantado y que no hay ningún servicio escuchando en el puerto 8888:

 [English](#)

## PÁGINAS

[Acerca de](#)  
[Autores](#)  
[Aviso Legal](#)  
[Colaboraciones](#)  
[Herramientas "self-made"](#)  
[Licencia de uso](#)  
[Política de cookies](#)  
[Política de privacidad](#)

## BUSCAR

Search this website ...

## LAB52

[Grupo Garamedon](#)  
[Grupo WIRTE](#)  
[Grupo ORANGEWORM](#)

## AUTORES

[Manuel Benet](#) (224)  
[Antonio Villalón](#) (218)  
[Antonio Sanz](#) (105)  
[Colaboradores](#) (84)  
[Maite Moreno](#) (73)  
[Roberto Amado](#) (61)  
[José Rosell](#) (59)  
[Joaquín Moreno](#) (54)  
[Josemi Holguín](#) (54)  
[Jose L. Villalón](#) (50)  
[Joan Soriano](#) (48)  
[Nelo Belda](#) (43)  
[José Vila](#) (43)  
[Antonio Huerta](#) (38)  
[Fernando Seco](#) (38)

[Todos los autores](#)

## ARCHIVOS

Elegir el mes

## BLOGS

[Bruce Schneier](#)  
[Dancho Danchev](#)  
[F-Secure](#)  
[FireEye Threat Research](#)

```
# netstat -putan | grep 8888
# lsof -Pi | grep 8888
```

A continuación vamos a emplear un pequeño script en bash, agradecer la ayuda a Raúl Rodríguez (no te escondas), que requiere tener instalado el binario **Nemesis** (permite manipular paquetes de forma sencilla). A dicho script le pasamos una orden entre comillas, de tal forma que generará un paquete por cada letra, donde el valor ID del campo IP, será el valor ASCII de la letra indicada.

Dentro del script tenemos la variable IP\_DST y DPORT donde hay que indicar la IP y puerto de nuestro servidor víctima. También tenemos las variables IP\_ORIGEN y SPORT donde se indica la IP y Puerto origen (puede ser cualquiera). Para el POC vamos a indicar a Nemesis que la dirección IP de origen es la de un servidor de Google y el puerto origen es el 80. De esta forma al servidor víctima le llegarán paquetes con el flag ACK con IP origen de Google y el puerto 80.

De este modo, si leemos la traza de red veremos únicamente que están llegando paquetes ACK de google, algo de lo más normal, cuando realmente lo que nos está llegando son comandos de una *backdoor*.

Para esta demo vamos a crear el usuario "ximo" (que me han dicho que es muy peligroso) en la máquina víctima:

Cliente:

[illegible]

Servidor:

```
# ./servidor -port 8888
Dato recibido: 117
Dato recibido: 115
Dato recibido: 101
Dato recibido: 114
Dato recibido: 97
Dato recibido: 100
Dato recibido: 100
Dato recibido: 32
Dato recibido: 120
Dato recibido: 105
Dato recibido: 109
Dato recibido: 111
Comando: useradd ximo
```

## ¿Habrá funcionado?

Security Clan House  
Steve Bellovin  
Tenable Security  
The Invisible Things  
Veracode Blog  
Vlan7  
Zero Day

## ORGANISMOS

CSIRT-CV  
INCIBE

## META

Acceder  
Feed de entradas  
Feed de comentarios  
WordPress.org

## Licencia de uso del contenido

#

Ahora vamos a ver la traza de red desde el servidor (IP 172.17.X.X) cuando se ha enviado el comando "useradd ximo". En este caso recordar que hemos usado la IP de Google, por lo que la IP origen usada es "66.249.92.104" obtenida de un simple ping:

```
$ ping -c1 google.es
PING google.es (66.249.92.104) 56(84) bytes of data.
```

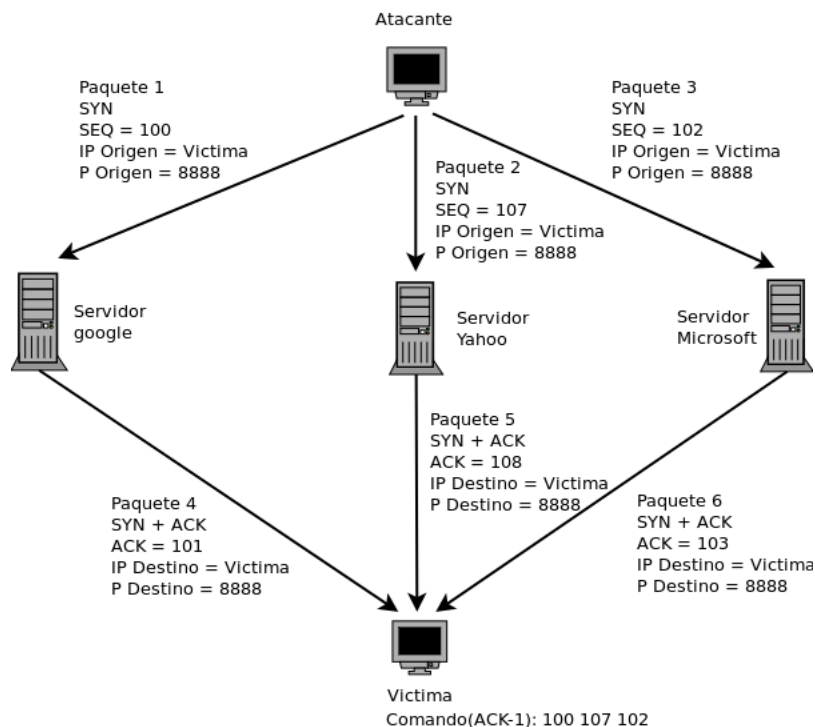
Veamos la traza:

```
# tcpdump -nn -vvv -i eth0 port 8888
10:00:55.265912 IP (tos 0x0, ttl 254, id 117, offset 0, flags [none], proto TCP (6),
  length 40)
  66.249.92.104.80 > 172.17.X.X.8888: Flags [.], cksum 0x2912 (correct),
  seq 283280788, ack 1457724121, win 4096, length 0
10:00:55.265964 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 40)
  172.17.X.X.8888 > 66.249.92.104.80: Flags [R], cksum 0xcf94 (correct), seq
  1457724121, win 0, length 0
10:00:55.270636 IP (tos 0x0, ttl 254, id 115, offset 0, flags [none], proto TCP (6),
  length 40)
  66.249.92.104.80 > 172.17.X.X.8888: Flags [.], cksum 0x2909 (correct), seq
  1069023501, ack 78166684, win 4096, length 0
10:00:55.270669 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 40)
  172.17.X.X.8888 > 66.249.92.104.80: Flags [R], cksum 0x1051 (correct), seq
  1535890804, win 0, length 0
10:00:55.280132 IP (tos 0x0, ttl 254, id 101, offset 0, flags [none], proto TCP (6),
  length 40)
  66.249.92.104.80 > 172.17.X.X.8888: Flags [.], cksum 0x827f (correct), seq
  1249543965, ack 4145208809, win 4096, length 0
10:00:55.280157 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 40)
  172.17.X.X.8888 > 66.249.92.104.80: Flags [R], cksum 0xfa99 (correct), seq
  1307965633, win 0, length 0
10:00:55.284502 IP (tos 0x0, ttl 254, id 114, offset 0, flags [none], proto TCP (6),
  length 40)
  66.249.92.104.80 > 172.17.X.X.8888: Flags [.], cksum 0x5a89 (correct), seq
  4108084880, ack 626806209, win 4096, length 0
```

Al servidor nos han llegado una serie de paquetes TCP con flag ACK (!!) desde la IP de Google (66.249.92.104), con puerto origen 80 sin *payload*, al cual hemos respondido con un paquete RST. Si nos fijamos en los paquetes de envío veremos el campo "id número", que identifica el valor del campo id del protocolo IP, el cual tiene valores como 117, 115, 101, 104, ... que corresponden con los valores ASCII de u, s, e, r, ...

Como podréis suponer, la posibilidad de rastrear ese comando es realmente compleja puesto que en ningún momento nos llega la IP del atacante ya que se falsifica la IP origen.

Pero vayamos más allá: ¿qué ocurriría si en vez de jugar con la cabecera ID del protocolo IP, usáramos el valor de secuencia (SEQ) del paquete TCP, de tal forma que enviáramos un paquete TCP con flag SYN (inicio de conexión) a un servidor público cuya IP y puerto origen sea la de la víctima? Pues que el servidor público contestaría a la víctima con un paquete SYN ACK al puerto indicado como puerto origen, cuyo valor ACK corresponderá al valor de sesión enviado por el atacante + 1. Por tanto se podría usar servidores públicos legítimos para que mandaran en la cabecera TCP la orden a ejecutar, siendo todavía más difícil rastrear quien ejecutó la orden. Veamos una imagen que aclara esta idea:



Esto último, así como posteriores pruebas, quedan como ejercicio para el lector. Los desarrollos realizados para la prueba de concepto, que hemos denominado **CovertShell**, están a disposición del público. Esperamos que les sea de ayuda, sin olvidar por supuesto que se ponen a disposición del público para fines educativos y de investigación.

## Comments



DavidLladro says

27 de octubre de 2010 at 9:14 am

Muy buen artículo Ximo, y gracias por la herramienta!!



JoseMiHolguin says

27 de octubre de 2010 at 9:54 am

Una gran entrada!

Un saludo.



Dani says

27 de octubre de 2010 at 10:31 am



ramado says

27 de octubre de 2010 at 10:33 am

Gran entrada!!!! hum esto puede tener muchas aplicaciones...



rrodriguez says

27 de octubre de 2010 at 11:14 am

Buen post Ximo!... enhorabuena.



maite says

27 de octubre de 2010 at 1:38 pm

Gran post Ximo!



nbelda says

27 de octubre de 2010 at 4:43 pm

Buena entrada, Ximo.

En el punto que hablas de utilizar señuelos y servidores de terceros para que realicen esa comunicación, poco se puede hacer. Lo mismo ocurre cuando se realizan ataques DDoS.

Respecto a la dificultad de descubrir este tipo de tráfico, creo que hay dos opciones y ninguna buena: esperar que el software sea reconocido y detectado por firma analizando las cabeceras TCP en un IDS, o buscando anomalías en la comunicación TCP pero esto puede provocar una gran cantidad de falsos positivos.

Me explico. El paquete con el flag ACK sólo debe recibirse después de establecer la comunicación, de hecho es el último paso, pero debe de ser muy difícil de descubrir aunque se detecte ese tráfico. Así, podría saltar alertas cuando se recibe un paquete ACK sin que se haya recibido previamente SYN y SYN-ACK, pero al utilizar el campo SEQ no se sabe a qué conversación corresponde, así que se podría aprovechar esto para detectarlo más fácilmente.

Es una idea que propongo para intentar "detectar" este tipo de ataques. ¿Creéis que podría ser útil para detectar este tipo de ataques?

Un saludo.



vierito5 says

27 de octubre de 2010 at 4:53 pm



FSeco says  
27 de octubre de 2010 at 7:31 pm

Chapeau Ximo.



pj says  
27 de octubre de 2010 at 11:25 pm

nbelda, cualquier firewall serio hoy en día comprueba las secuencias TCP y sería tan trivial para esto como simplemente no hacer nada. De todas maneras a mi personalmente me ha encantado este artículo, gracias!



Samuel Segarra says  
28 de octubre de 2010 at 10:10 am

Gracias por la aportación Ximo, una entrada muy interesante.



JoseMi says  
1 de junio de 2011 at 1:39 pm

LOKI2  
--

LOKI2 is an information-tunneling program

<http://www.phrack.org/issues.html?issue=51&id=6>



tano says  
20 de mayo de 2015 at 10:09 pm

Hola, quisiera consultar sobre la herramienta covert shell, donde se puede descargar, ya que debo hacer un trabajo de grado en la facultad y hay que exponer la prueba de concepto con esa herramienta.  
Muchas gracias.

## Trackbacks

de la red - 27/10/2010 « Tecnologías y su contexto dice:  
28 de octubre de 2010 a las 2:32 am

el protocolo TCP/IP, y proporcionaremos una herramienta, CovertShell, diseñada para la prueba de concepto. Pueden localizar los fuentes al final de la entrada. [...]

[Return to top of page](#)

Copyright © 2025 · Genesis Framework · WordPre