CACIC2007
XIII Congreso Argentino de Ciencias de la Computación

Mejorando la Resistencia de TCP a Ataques Ciegos mediante Aleatorización de Puertos Efímeros

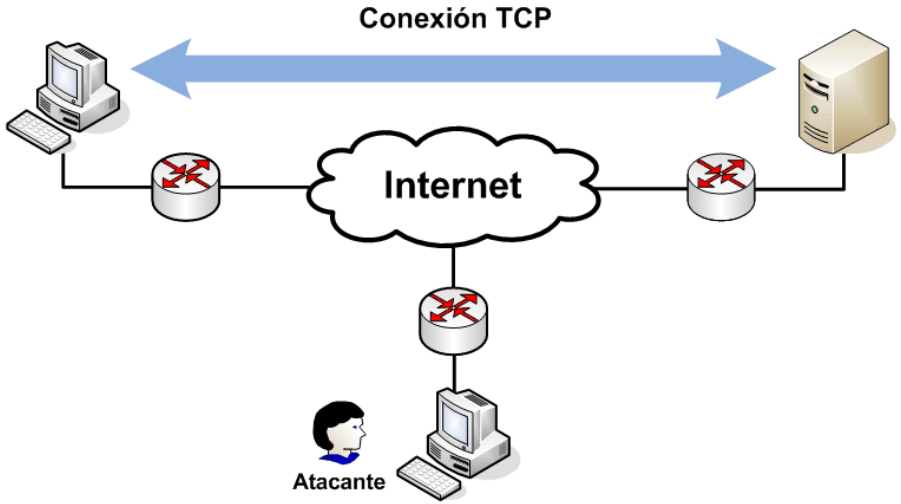
Fernando Gont
UTN/FRH, Argentina

II Workshop de Arquitecturas, Redes y Sistemas Operativos
CACIC 2007

2 al 5 de Octubre de 2007, Corrientes, Argentina

Ataques ciegos contra TCP

- En los últimos años se han divulgado dos familias de ataques “ciegos” contra TCP
  - “**Slipping in the Window**”: Ataques basados en segmentos TCP falsificados (NISCC vulnerability advisory #236929)
  - “**ICMP attacks against TCP**” : Ataques basados en paquetes ICMP falsificados (NISCC vulnerability advisory #532967)
- Estos ataques pueden ser realizados sin la necesidad de acceder a los paquetes pertenecientes a la conexión atacada.



- Estos ataques requieren que el atacante conozca o pueda adivinar el connection-id (client IP, client port, server IP, server port).

Aleatorización de puertos TCP

- Resulta lógico intentar mitigar la realización de ataques ciegos dificultando la habilidad del atacante de adivinar los valores {client IP, client TCP port, server IP, server TCP port}.
- De estos valores, el único valor que en principio es posible variar arbitrariamente es el client port.
- Sin embargo, en este aspecto la maýoría de las implementaciones facilitan la tarea del atacante más de lo necesario.
  - Utilizan para los puertos efímeros un rango de puertos muy reducido
  - El algoritmo de selección de puertos efímeros produce una secuencia trivialmente predecible (Algoritmo #1)

Selección de puertos efímeros

Algoritmos de selección de puertos efímeros

- Al seleccionar un puerto efímero, debe asegurarse que el connection-id resultante (client address, client port, server address, server port) no esté actualmente en uso.
- Si existe en el sistema local una conexión activa con dicho connection-id, se procederá a seleccionar otro puerto efímero, con el fin de salvar el conflicto.
- Sin embargo, es imposible para el sistema local poder detectar si existe alguna instancia de comunicación activa en el **sistema remoto** utilizando dicho connection-id (por ejemplo, unaconexión TCP en el estado TIME-WAIT).
- En caso que el puerto efímero seleccionado resultara en un connection-id en uso en el sistema remoto, se dice que se ha producido una “colisión de puertos”.
- Las colisiones de puertos pueden resultar en corrupción de información o bien fallos en el establecimiento de conexión. Por ello, deben ser evitadas.
- En consecuencia, la frecuencia de reuso de puertos debe ser minimizada.

Rango de puertos efímeros

IANA asigna a los puertos efímeros el rango 49152-65535. Sin embargo, la mayoría de los sistemas eligen sus “puertos efímeros” de un subespacio de todo el espacio de puertos disponible que, en la mayor[ia de los casos, difiere de aquél elegido por la IANA.

Sistema operativo	Puertos efímeros
Microsoft Windows	1024 - 4999
Linux kernel 2.6	1024 - 4999
Solaris	32768 - 65535
AIX	32768 - 65535
FreeBSD	1024 - 5000
NetBSD	49152 - 65535
OpenBSD	49152 - 65535

Secuencia generada por el algoritmo tradicional BSD (Algoritmo #1)

Nr.	IP:port	min_ephemeral	max_ephemeral	next_ephemeral	port
#1	128.0.0.1:80	1024	65535	1024	1024
#2	128.0.0.1:80	1024	65535	1025	1025
#3	170.210.0.1:80	1024	65535	1026	1026
#4	170.210.0.1:80	1024	65535	1027	1027
#5	128.0.0.1:80	1024	65535	1028	1028

Características del Algoritmo #1

- Es el implementado en la gran mayoría de las pilas TCP/IP
- Es simple
- Posee una frecuencia de reuso de puertos aceptable (aunque mayor que la necesaria)
- **Produce una secuencia de puertos efímeros trivialmente predecible**.

Aleatorización de puertos

Características de un buen algoritmo de aleatorización de puertos

- Minimizar la predictibilidad de los números de puerto utilizados para futuras conexiones salientes.
- Minimizar la frecuencia de reuso de puertos (es decir, evitar “colisiones”).
- Evitar conflictos con aplicaciones que dependen de la utilización de números de puertos específicos (por ejemplo, evitar utilizar para los puertos efímeros números de puerto como el 80, el 100, el 6667, etc.)

Rango de puertos efímeros

- Se debería utilizar para los puertos efímeros el rango 1024-65535
- Para evitar la utilización de determinados puertos, se podría utilizar un array de bits que permita indicar, para el mencionado rango, cuáles puertos se deben utilizar, y cuáles no.
- Esto, en conjunto con un buen algoritmo de aleatorización de puertos, dificulta la tarea del atacante de “adivinar” los puertos efímeros utilizados por el cliente.

Algoritmo de aleatorización básico (Algoritmo #2)

- Los puertos efímeros se seleccionan como un valor `random()` dentro del rango de puertos escogido para los puertos efímeros.

```
next_ephemeral = min_ephemeral + random()
                 % (max_ephemeral - min_ephemeral + 1)
```

Características del Algoritmo #2

- Ha sido implementado en OpenBSD y FreeBSD
- Produce una secuencia de puertos efímeros muy difícil de predecir
- La frecuencia de reuso de puertos puede ser mucho mayor que la del algoritmo BSD (Algoritmo #1)
- Usuarios de los mencionados sistemas operativos han reportado problemas de interoperatividad. En consecuencia, FreeBSD incorporó un hack que deshabilita la aleatorización de puertos cuando el número de conexiones salientes por unidad de tiempo excede un determinado valor.

Algoritmo de aleatorización avanzado (Algoritmo #3)

- Selecciona lospuertos efímeros como resultado de la función:
  - port = next_ephemeral + F(local IP, remote IP, remote port, secret-key)
  - next_ephemeral: es un contador que se incrementa por cada puertoefímero seleccionado
  - F: Funcion de hash
- Esta función separa el “espacio de puertos”, produciendo una secuencia incremental **distinta** para cada grupo de valores {local IP, remote IP, remote port}.
- El origen de esta secuencia es aleatorio, dependiendo el mismo de la calidad de la función de hash.

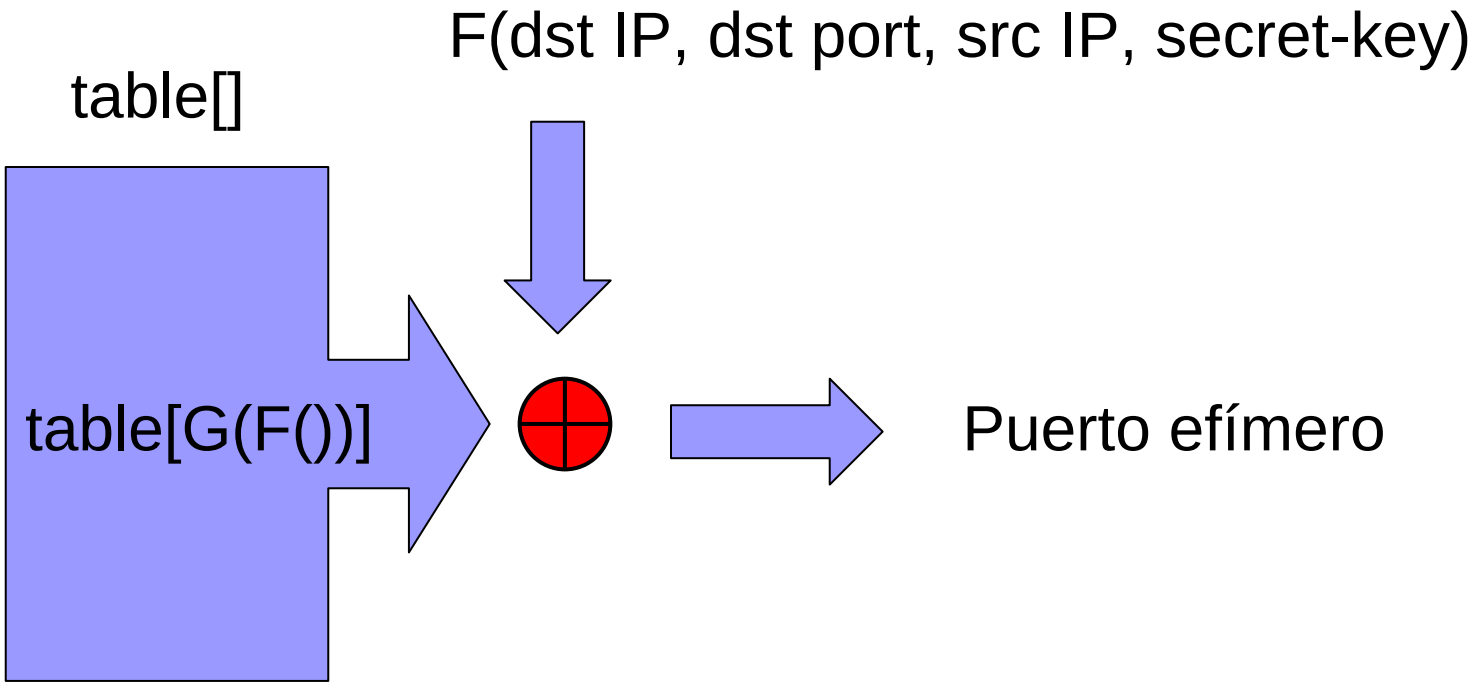
Secuencia producida por el Algoritmo #3

Nr.	IP:port	offset	min_ephemeral	max_ephemeral	next_ephemeral	port
#1	128.0.0.1:80	1000	1024	65535	1024	3048
#2	128.0.0.1:80	1000	1024	65535	1025	3049
#3	170.210.0.1:80	4500	1024	65535	1026	6550
#4	170.210.0.1:80	4500	1024	65535	1027	6551
#5	128.0.0.1:80	1000	1024	65535	1028	3052

Características del Algoritmo #3

- Implementado en el Linux Kernel
- Produce una secuencia muy difícil de predecir por terceros
- Posee una frecuencia de reuso de puertos igual a la del algoritmo tradicional BSD (Algoritmo #1)
- Su implementación es más compleja que la de los algoritmos anteriores

Algoritmo mejorado (Algoritmo #4)



- Se reduce la frecuencia de reuso de puertos mediante la separación del espacio de incrementos

Secuencia producida por el Algoritmo #4

Nr.	IP:port	offset	min_ephemeral	max_ephemeral	index	table[index]	port
#1	128.0.0.1:80	1000	1024	65535	10	1024	3048
#2	128.0.0.1:80	1000	1024	65535	10	1025	3049
#3	170.210.0.1:80	4500	1024	65535	15	1024	6548
#4	170.210.0.1:80	4500	1024	65535	15	1025	6549
#5	128.0.0.1:80	1000	1024	65535	10	1026	3050

Características del Algoritmo #4

- Se está trabajando en implementaciones apra FreeBSD y OpenBSD
- Produce una secuencia muy difícil de predecir por terceros
- Posee una frecuencia de reuso menor que la del algoritmo tradicional BSD (Algoritmo #1)
- Su implementación es más compleja que la de todos los algoritmos anteriores

Conclusiones

- La aleatorización de puertos es una medida proactiva para evitar ataques “ciegos” contra protocolos de transporte.
- Se puede aplicar a todos los protocolos de transporte existentes: TCP, UDP, SCTP, DCCP, etc.
- Lamentablemente, pocas implementaciones aleatorizan los puertos efímeros.
- De aquellas que lo hacen,muchas lo hacen incorrectamente, causando problemas de interoperabilidad.

Preguntas?

Información de contacto

Fernando Gont

fernando@gont.com.ar

Más información en:

http://www.gont.com.ar