



Intercepción de tráfico en Sistemas GNU/Linux

Jon Ander Ortiz

-Jonan-

jonbaine@gmail.com



Marzo 2006 - eghost

ESIDE – Universidad de Deusto



This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/2.0/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305,

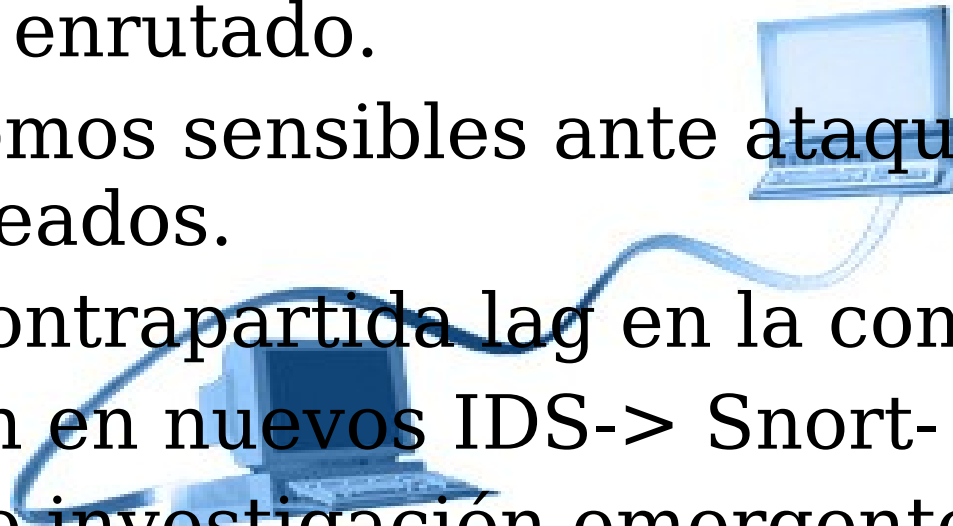




IMPORTANCIA DE LA INTERCEPCION DE TRÁFICO

- Análisis del tráfico antes de su enrutamiento.
- Tradicionalmente uso libpcap.
- Fallos en la concepción (copia de paquetes de la red):
 - Debilidades ante ataques spoofeados.
 - Fallos en la respuesta activa: Snort Sam.

IMPORTANCIA DE LA INTERCEPCION DE TRÁFICO II

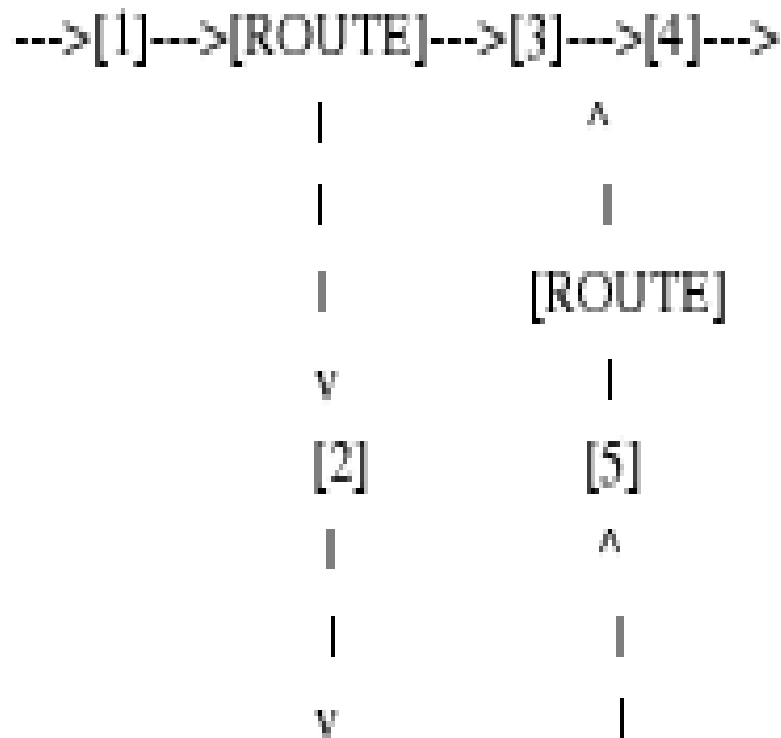
- En la intercepción:
 - Todos los paquetes son analizados antes de su enrutado.
 - No somos sensibles ante ataques spoofeados.
 - Como contrapartida lag en la comunicación
 - Solución en nuevos IDS-> Snort- Inline.
 - Línea de investigación emergente.
- 

GESTIÓN DE TRÁFICO GNU/LINUX

- Siempre ha sido un tema importante en GNU/Linux.
- Hoy en día Netfilter (desde kernel-2.4)
 - Creado por Rusty Rusell (Creador también de ipchains)
 - Arquitectura C/S (evolución de ipchains).
 - ipchains--> bastantes problemas:
 - difícil programar
 - cancer montar proxys...
 - No era posible hacer reglas independientes de interfaz
 - No había manera estándar pasar tráfico a user



GESTIÓN TRÁFICO GNU/LINUX II



- Conjunto de ganchos en las pilas de protocolos (en varias pilas -> ipv4, ipv6...)

- NF_IP_PRE_ROUTING [1]
- NF_IP_LOCAL_IN [2]
- NF_IP_FORWARD [3]
- NF_IP_POST_ROUTING[4]
- NF_IP_LOCAL_OUT[5]

GESTIÓN TRÁFICO GNU/LINUX - III

- Los módulos del kernel pueden hookearse a estos ganchos y ser llamados por el kernel, cuando llegue un paquete.
- Las iptables -> Compuesto por 3 tablas (filter, NAT, mangle)
- Filtros en la tabla filter.
- El filtro en los ganchos:
 - NF_IP_LOCAL_IN
 - NF_IP_FORWARD
 - NF_IP_LOCAL_OUT

PRIMERA APROX. -> KERNEL :D

- La primera aproximación módulo del kernel que filtre.
- Adecuado con grandes flujos de tráfico.
- Utilizamos la API de Netfilter en Kernel para interceptar tráfico.
- EJEMPLO:
 - Modificación de todos los ICMP que salen de nuestro sistema :D

PRIMERA APROX -> KERNEL :d II

- PODEMOS :
 - NF ACCEPT: aceptar.
 - NF DROP: tirar.
 - NF STOLEN: nos ocupamos nosotros
 - NF QUEUE: encolamos para pasar a userland.
 - NF REPEAT: repetimos la llamada al gancho -> ojo con interbloqueos.



libiptc

- Librería para manipular iptables desde userland :D.
- Necesitamos root.
- Internamente hace set/get sockopt. De esta manera configuramos internamente las iptables.
- EJEMPLO : Listado de las cadenas de la tabla filter.
- Mucho más extensible... da juego... a ver quien se anima. :D

NETLINK sockets kernel - userland

- Sistema de sockets para comunicación kernel- userland mediante dispositivos virtuales.
- La comunicación recuerda a sockets Berkley.
- Necesita:
 - Tener el kernel con soporte Netlink.
 - Tener el kernel recompilado si queremos hacer comunicación nueva.



NETLINK sockets kernel – userland II

- Sockets especiales, no tiene puertos, tiene protocolos.
- Soporta comunicación multicast.
- Lo que hay en el kernel en principio es unos cuantos de protocolos. Hay que conocerlo para implementarlo:
 - `#define NETLINK_ROUTE 0 /* Routing/devicehook*/`
 - `#define NETLINK_SKIP 1 /* Reserved for ENskip*/`
 - `#define NETLINK_FIREWALL 3 /*Firewall hook*/`
- Si queremos uno nuestro -> recompilar kernel con otro nuevo



NETLINK sockets kernel-userland III

- Para en ejemplo se ha creado en protocolo `NETLINK_MENDIZALE` :D
- Ejemplo -> Comunicación unicast user-kernel. :D
- Tiene alguna ventaja sobre `copy_to_user` ->
 - * No necesitamos saber el puntero.
 - * No tiene pq haber nadie escuchando.

Bastante complicado -> WRAPPER LIBIPQ



LIBIPQ, INTERCEPCION EN USERLAND

- Muy poco conocida.
- Necesita iptables-dev
- Wrapper a nivel de userland del protocolo NETLINK_FIREWALL. Una abstraccion bastante wapa.
- Provee de un API a nivel de userland para escuchar paquetes que nos han econlado desde kernel.



LIBIPQ, INTERCEPCION EN USERLAND

- Necesitamos que alguien nos encole en kernel.
 - Bien un módulo del Kernel, enganchado a Netfilter.
 - Bien una regla de IPTABLES:
 - `# iptables -A OUTPUT -p tcp -j QUEUE`
- Ejemplo: Pillamos los icmp's.
- Nos permite modificar tráfico que vamos a enrutar en userland :D.

AMAITU DA :D



MILA ESKER ETORTZEAGATIK.
MUCHAS GRACIAS POR VENIR.