

TCPdump, Librairie PCAP

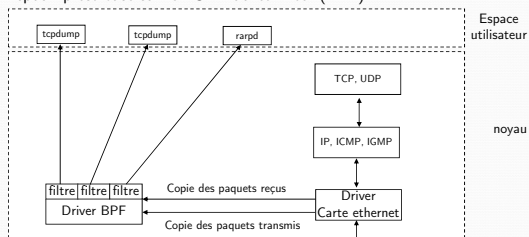
Stéphane CATELOIN



v2.1/2019.09

TCPdump

- Place la carte réseau en **mode confus** (*promiscuous mode*)
- Chaque paquet circulant sur le lien est capturé
- Tcpcdump est basé sur le BSD Packet Filter (BPF)



Filtre BPF

- Les filtres spécifiés par l'utilisateur indiquent à BPF quelles trames sont jugées intéressantes
- En filtrant au niveau du noyau et pas au niveau utilisateur, le volume de données échangé noyau/process est minimisé
- Exemples d'expressions tcpdump définissant des filtres


```
tcpdump tcp port 21
tcpdump 'icmp[0] !=8 and icmp[0] !=0' (ni echo, ni reply)
```

Affichage de tcpdump

- Tcpdump fournit une sortie brute des données

```
11:59:17.894013 api-cisco.u-strasbg.fr > clarinet.u-strasbg.fr: icmp: time
exceeded in-transit
4500 0038 e04b 0000 ff01 4186 824f 49fe
824f 4b56 0b00 aa8c 0000 0000 4500 0028
ddb5 0000 0111 7e16 824f 4b56 824f 0e05
ddb4 829b 0014
11:59:17.900110 api-cisco.u-strasbg.fr > clarinet.u-strasbg.fr: icmp: time
exceeded in-transit
4500 0038 e04c 0000 ff01 4186 824f 49fe
824f 4b56 0b00 a0a8 0000 0000 4500 0028
ddb6 0000 0111 7e15 824f 4b56 824f 0e05
ddb4 829c 0014
```

Affichage tcpdump

- La sortie tcpdump affiche
 - l'estampille horaire (11:59:17.894013)
 - le nom de la machine émettrice (api-cisco.u-strasbg.fr)
 - le nom de la machine destination (clarinet.u-strasbg.fr)
 - le protocole utilisé et la signification du message (icmp: time exceeded in-transit)
 - le dump du paquet

Tcpdump et la sécurité

- Tcpdump vous permet d'analyser n'importe quel paquet qui circule sur le lien
- Par défaut, seul root est autorisé à analyser les paquets en mode confus

tcpdumpx

```
# tcpdump -lnx -s 1500 -i ./tcpdumpnx

:58:30.459994 178.147.72.193.2898 > 127.0.0.1.80: tcp 30 [ttl 1]

IP_HDR=20 IP_OPT=0 TCP_HDR=20 TCP_OPT=0 DATA=30 FLAGS= PUSH ACK

IP_HDR    45    00    00    46    14    16    00    00    01    06
whl tos len id id off off ttl pro
IP_HDR    22    47    b2    93    48    c1    7f    00    00    01
sum sum src arc src arc dst dat dat
TCP_HDR    06    52    00    50    4f    ee    06    a7    3a    d1
src src dst dst seq seq seq seq ack
TCP_HDR    bb    00    50    18    02    00    4c    9d    00    00
ack ack off flag win win sum sum urp urp
DATA       47    45    54    20    2f    63    67    69    2a    62
6   E   T /   6   8   1   -   b
DATA       69    6e    2f    70    65    72    6c    2e    65    78
i   n   /   p   e   r   l   .   e   x
DATA       6e    20    48    54    54    50    2f    31    2e    30
e   H   T   T   P   /   1   .   0
```

ethereal (wireshark)

demo.pcap - ethereal						
#	Time	Source	Destination	Protocol	Info	Length
0	0.000000	192.168.0.1	192.168.0.1	ICMP	0.000000 192.168.0.1: 192.168.0.1	60
1	2.000100	192.168.0.1	192.168.0.1	ICMP	0.000100 192.168.0.1: 192.168.0.1	60
2	2.000100	192.168.0.1	192.168.0.1	ICMP	0.000100 192.168.0.1: 192.168.0.1	60
3	4.000100	192.168.0.1	192.168.0.1	ICMP	0.000100 192.168.0.1: 192.168.0.1	60
4	6.000200	192.168.0.1	192.168.0.1	ICMP	0.000200 192.168.0.1: 192.168.0.1	60
5	6.000200	192.168.0.1	192.168.0.1	ICMP	0.000200 192.168.0.1: 192.168.0.1	60
6	8.000300	192.168.0.1	192.168.0.1	ICMP	0.000300 192.168.0.1: 192.168.0.1	60
7	8.000300	192.168.0.1	192.168.0.1	ICMP	0.000300 192.168.0.1: 192.168.0.1	60
8	10.000400	192.168.0.1	192.168.0.1	ICMP	0.000400 192.168.0.1: 192.168.0.1	60
9	10.000400	192.168.0.1	192.168.0.1	ICMP	0.000400 192.168.0.1: 192.168.0.1	60
10	12.000500	192.168.0.1	192.168.0.1	ICMP	0.000500 192.168.0.1: 192.168.0.1	60
11	12.000500	192.168.0.1	192.168.0.1	ICMP	0.000500 192.168.0.1: 192.168.0.1	60
12	14.000600	192.168.0.1	192.168.0.1	ICMP	0.000600 192.168.0.1: 192.168.0.1	60
13	14.000600	192.168.0.1	192.168.0.1	ICMP	0.000600 192.168.0.1: 192.168.0.1	60
14	16.000700	192.168.0.1	192.168.0.1	ICMP	0.000700 192.168.0.1: 192.168.0.1	60
15	16.000700	192.168.0.1	192.168.0.1	ICMP	0.000700 192.168.0.1: 192.168.0.1	60
16	18.000800	192.168.0.1	192.168.0.1	ICMP	0.000800 192.168.0.1: 192.168.0.1	60
17	18.000800	192.168.0.1	192.168.0.1	ICMP	0.000800 192.168.0.1: 192.168.0.1	60
18	20.000900	192.168.0.1	192.168.0.1	ICMP	0.000900 192.168.0.1: 192.168.0.1	60
19	20.000900	192.168.0.1	192.168.0.1	ICMP	0.000900 192.168.0.1: 192.168.0.1	60
20	22.001000	192.168.0.1	192.168.0.1	ICMP	0.001000 192.168.0.1: 192.168.0.1	60
21	22.001000	192.168.0.1	192.168.0.1	ICMP	0.001000 192.168.0.1: 192.168.0.1	60
22	24.001100	192.168.0.1	192.168.0.1	ICMP	0.001100 192.168.0.1: 192.168.0.1	60
23	24.001100	192.168.0.1	192.168.0.1	ICMP	0.001100 192.168.0.1: 192.168.0.1	60
24	26.001200	192.168.0.1	192.168.0.1	ICMP	0.001200 192.168.0.1: 192.168.0.1	60
25	26.001200	192.168.0.1	192.168.0.1	ICMP	0.001200 192.168.0.1: 192.168.0.1	60
26	28.001300	192.168.0.1	192.168.0.1	ICMP	0.001300 192.168.0.1: 192.168.0.1	60
27	28.001300	192.168.0.1	192.168.0.1	ICMP	0.001300 192.168.0.1: 192.168.0.1	60
28	30.001400	192.168.0.1	192.168.0.1	ICMP	0.001400 192.168.0.1: 192.168.0.1	60
29	30.001400	192.168.0.1	192.168.0.1	ICMP	0.001400 192.168.0.1: 192.168.0.1	60
30	32.001500	192.168.0.1	192.168.0.1	ICMP	0.001500 192.168.0.1: 192.168.0.1	60
31	32.001500	192.168.0.1	192.168.0.1	ICMP	0.001500 192.168.0.1: 192.168.0.1	60
32	34.001600	192.168.0.1	192.168.0.1	ICMP	0.001600 192.168.0.1: 192.168.0.1	60
33	34.001600	192.168.0.1	192.168.0.1	ICMP	0.001600 192.168.0.1: 192.168.0.1	60

Frame 15 (156 on wire, 154 captured)

Internet Protocol, Src Addr: 192.168.0.1 (En.08:00:2B:01:02:01), Dest Addr: 192.168.0.1 (En.08:00:2B:01:02:01), Src Port: 200 (UDP), Dest Port: 200 (UDP), Seq: 199999999, Len: 60

4779721: 200 0x0000

Checksum: 0x0000, Seq: 199999999, Offset: 0

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

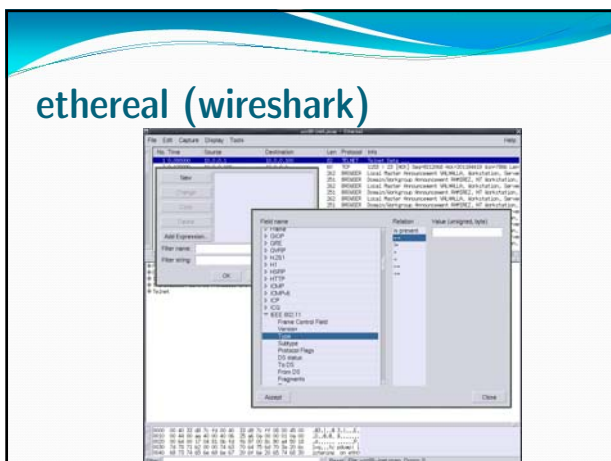
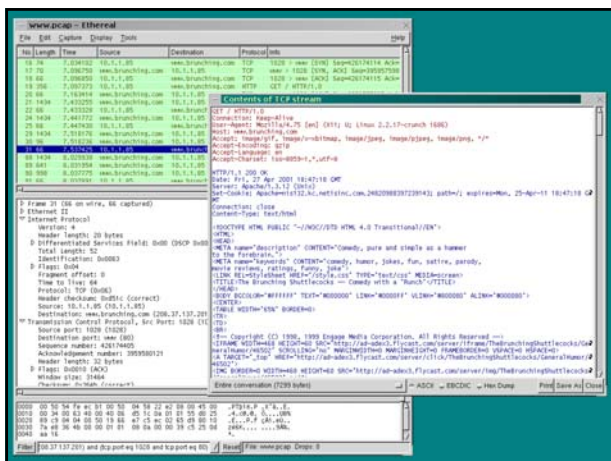
Header: 4779721: 200 0x0000

Header: 4779721: 200 0x0000

Header: 4

ethereal (wireshark)

[illegible]



Installation

- Téléchargement www.tcpdump.org
- Installation habituelle...
 - `./configure`
 - `make`
 - `make install`

Descriptions des fonctions & procédures

- `char* pcap_lookupdev (char* errbuf)`
 - recherche de l'interface
 - retourne une chaîne de caractères correspondant à l'interface
 - NULL sinon

Descriptions des fonctions & procédures

- `int pcap_lookupnet (char* device, bpf_u_int32 *netaddr, bpf_u_int32 *netmask, char* errbuf)`
 - recherche de l'adresse IP et du masque de sous réseau
 - retourne 0 si recherche fructueuse, -1 sinon
- (voir man `inet_ntoa`)

Descriptions des fonctions & procédures

- **pcap_t* pcap_open_live(**
char* device,
int snaplen,
int promisc,
int to_ms,
char* errbuf)
 - ouverture de la capture
 - device : nom de l'interface
 - snaplen : nb max d'octets capturés / trame
 - promisc : 1/0 mode confus
 - to_ms : timeout

Descriptions des fonctions & procédures

- **pcap_t* pcap_open_offline (const char *fname,**
char *errbuf)
 - ouverture d'une capture sur fichier
 - fname : fichier capture tcpdump
- **void pcap_close(pcap_t *p)**
 - fermeture de la capture

Descriptions des fonctions & procédures

- Gestion des filtres (man tcpdump)
 - **int pcap_compile(pcap_t *p, struct bpf_program**
***fp, char *str, int optimize, bpf_u_int32 netmask)**
 - compilation du filtre
 - p : descripteur capture
 - bpf_program : contiendra le filtre compilé
 - str : filtre
 - optimize : inutilisé
 - netmask : masque sous-réseau
 - retourne 0 si OK, -1 sinon

Descriptions des fonctions & procédures

- Gestion des filtres (suite)
 - **int pcap_setfilter(pcap_t *p, struct bpf_program *fp)**
 - associe le filtre compilé à la capture

Fonction de capture

- **int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user)**
 - capture en boucle
 - p : descripteur de la capture
 - cnt : nb de paquets à capturer (-1 : infini)
 - callback : nom de la fonction de callback
 - user : paramètres additionnels pour le callback

Callback

- Prototype de la fonction de callback: **void got_packet(u_char *args, const struct pcap_pkthdr *header, const u_char *packet);**
 - appelée à chaque capture de paquet
 - args : paramètre additionnel (dernier argument de pcap_loop)
 - header : informations sur la paquet
 - heure
 - taille...
 - packet : pointeur sur le paquet capturé

Analyse

- Utilisation des structures d'en-têtes
 - net/ethernet.h
 - netinet/ip.h
 - netinet/tcp.h
 - ...
 - (bootp.h)

Analyse

- exemple :


```
struct ether_header {
    u_char ether_dhost[ETHER_ADDR_LEN]; /*Destination host address*/
    u_char ether_shost[ETHER_ADDR_LEN]; /* Source host address */
    u_short ether_type; /* IP? ARP? RARP? etc */
};
```
- but : « calquer » le paquet capturé sur la structure


```
const struct ether_header *ethernet;
const struct iphdr *ip;
int size_ethernet = sizeof(struct ether_header);
ethernet = (struct ether_header*)(packet);
ip = (struct iphdr*)(packet + size_ethernet);
...
```
