

# Activité : mise en oeuvre réseau de la caméra IP

Thierry Vaira <[tvaira@free.fr](mailto:tvaira@free.fr)>

07/12/2015 (rev. 1)

## Table des matières

<b>Mise en oeuvre réseau de la caméra IP</b>	<b>1</b>
Adressage IP par DHCP . . . . .	1
Services disponibles . . . . .	5
Accès web : test en ligne de commande . . . . .	6
Accès <code>telnet</code> . . . . .	10

## Mise en oeuvre réseau de la caméra IP

### Adressage IP par DHCP

Pour communiquer avec la caméra IP, il faut que celle-ci possède une adresse IP dans le réseau local.

On distingue deux situations pour assigner une adresse IP à un équipement :

- de manière statique : l'adresse est fixe et configurée le plus souvent manuellement puis stockée dans la configuration de son système d'exploitation.
- de manière dynamique : l'adresse est automatiquement transmise et assignée grâce au protocole DHCP (*Dynamic Host Configuration Protocol*) ou BOOTP.

Dans notre situation, l'adresse IP de la caméra est assignée par un serveur DHCP.

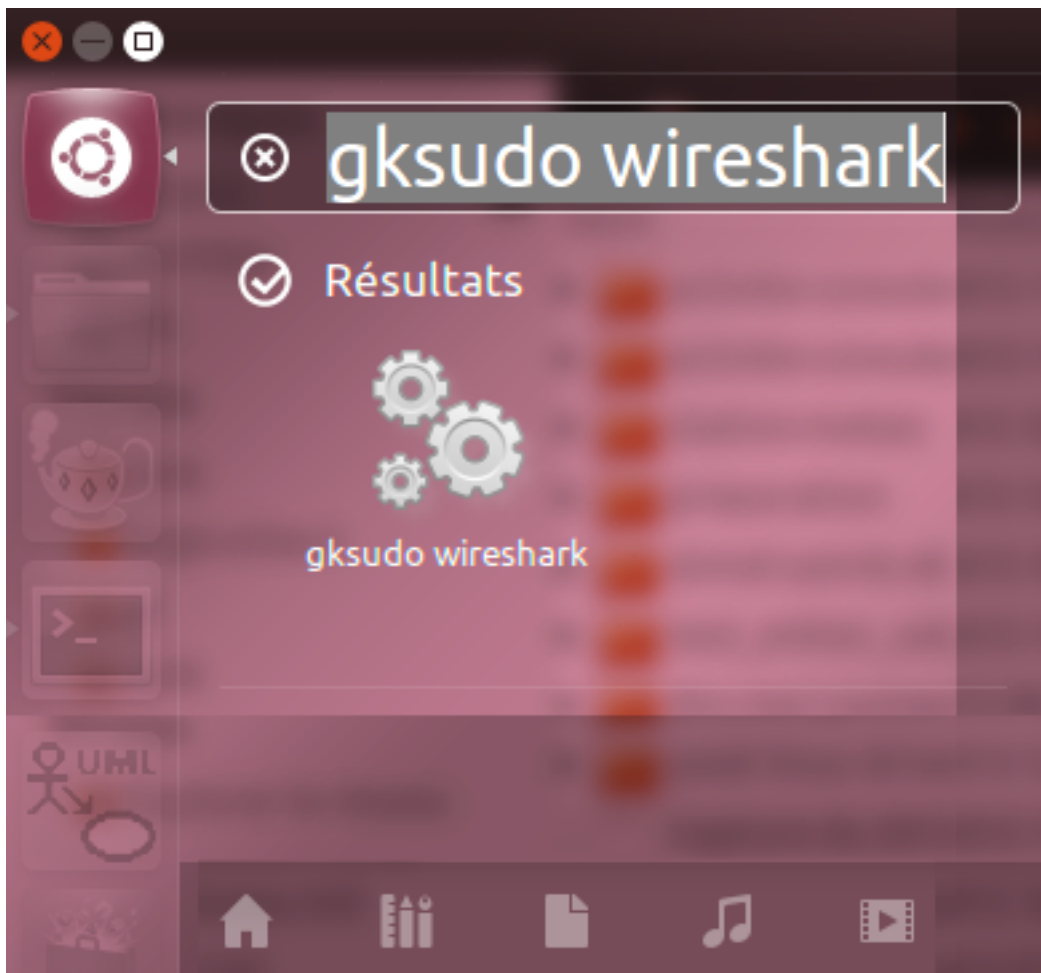
Il vous faut connaître l'adresse MAC de votre caméra.

Une adresse MAC (Media Access Control address) est un identifiant physique stocké dans une carte réseau ou une interface réseau et utilisé pour attribuer mondialement une adresse unique (codé sur 48 bits). L'adresse MAC est utilisée dans les trames transmises. Une trame transporte un paquet. L'adresse MAC identifie donc de manière unique l'interface physique de communication. Par exemple, on aura donc une adresse MAC pour l'interface Ethernet et une autre pour l'interface Wifi.

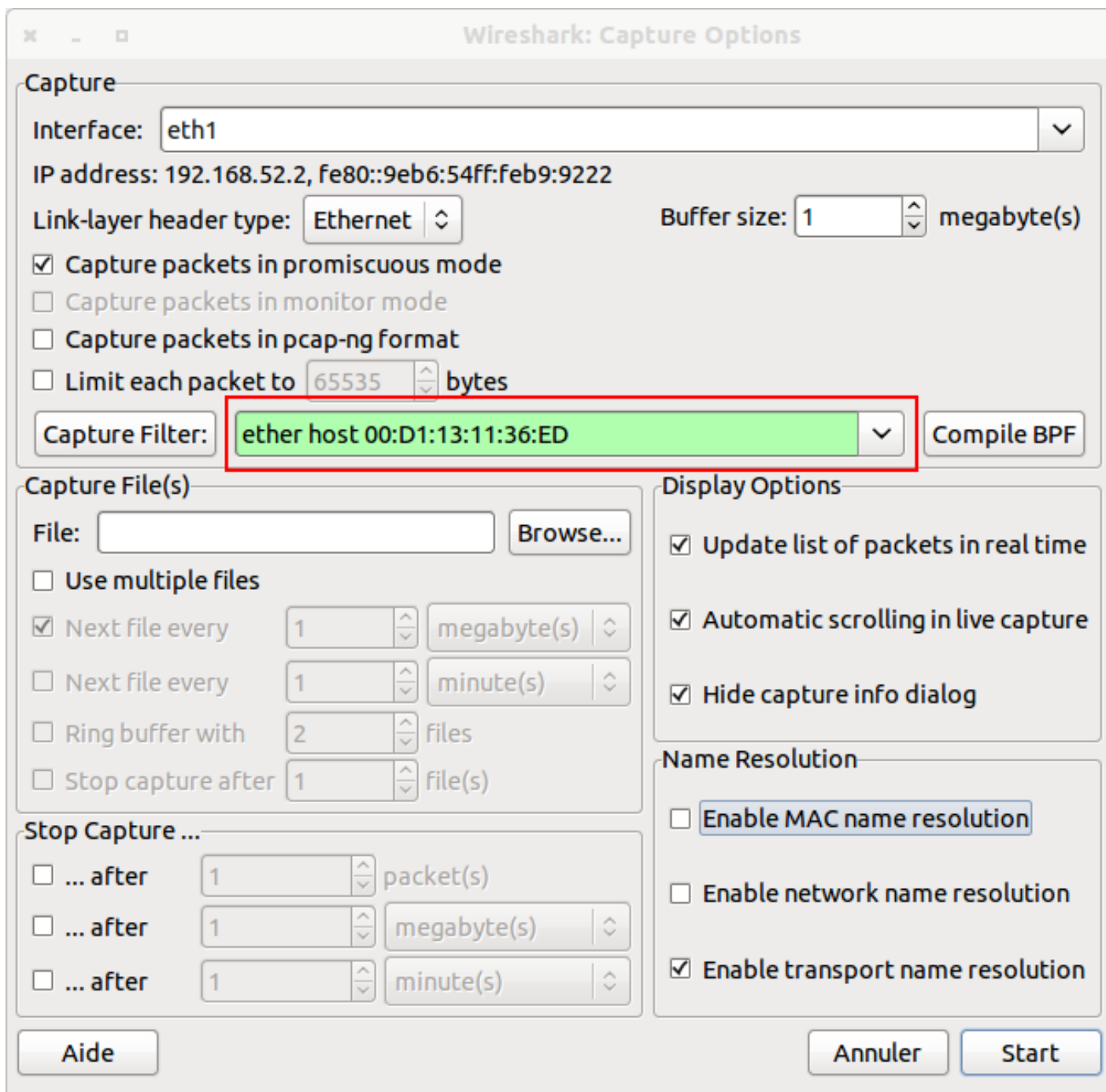
Pour cette partie, vous allez avoir besoin d'un analyseur de protocoles. Lancer Wireshark en mode root.

```
$ gksudo wireshark &
```

Ou :



On va capturer le trafic en filtrant celui en provenance de l'adresse MAC de la caméra :



The image shows the 'Wireshark: Capture Options' dialog box. The 'Capture' section is highlighted with a red box, showing the 'Capture Filter' set to 'ether host 00:D1:13:11:36:ED'. Other options include 'Interface: eth1', 'IP address: 192.168.52.2, fe80::9eb6:54ff:feb9:9222', 'Link-layer header type: Ethernet', 'Buffer size: 1 megabyte(s)', and checkboxes for 'Capture packets in promiscuous mode', 'Capture packets in monitor mode', 'Capture packets in pcap-ng format', and 'Limit each packet to 65535 bytes'. The 'Capture File(s)' section includes options for file naming and storage. The 'Display Options' section has checkboxes for 'Update list of packets in real time', 'Automatic scrolling in live capture', and 'Hide capture info dialog'. The 'Name Resolution' section has checkboxes for 'Enable MAC name resolution', 'Enable network name resolution', and 'Enable transport name resolution'. Buttons for 'Aide', 'Annuler', and 'Start' are at the bottom.

**Wireshark: Capture Options**

**Capture**

Interface: eth1  
IP address: 192.168.52.2, fe80::9eb6:54ff:feb9:9222  
Link-layer header type: Ethernet  
Buffer size: 1 megabyte(s)

Capture packets in promiscuous mode  
 Capture packets in monitor mode  
 Capture packets in pcap-ng format  
 Limit each packet to 65535 bytes

Capture Filter: ether host 00:D1:13:11:36:ED **Compile BPF**

**Capture File(s)**

File:  **Browse...**

Use multiple files

Next file every 1 megabyte(s)  
 Next file every 1 minute(s)  
 Ring buffer with 2 files  
 Stop capture after 1 file(s)

**Stop Capture ...**

... after 1 packet(s)  
 ... after 1 megabyte(s)  
 ... after 1 minute(s)

**Display Options**

Update list of packets in real time  
 Automatic scrolling in live capture  
 Hide capture info dialog

**Name Resolution**

Enable MAC name resolution  
 Enable network name resolution  
 Enable transport name resolution

**Aide** **Annuler** **Start**

Si vous êtes relié à un **switch**, vous ne pourrez seulement capturer le trafic émis par la caméra en broadcast.  
En effet, le **switch** se comporte comme un filtre au niveau des adresses MAC.

On démarre la capture puis on allume la caméra :

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	fe80::2d1:13ff:fe11:36ed	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
2	1.449040	fe80::2d1:13ff:fe11:36ed	ff02::2	ICMPv6	70	Router Solicitation from 00:d1:13:11:36:ed
3	3.178563	0.0.0.0	255.255.255.255	UDP	64	Source port: 6808 Destination port: 6809
4	3.189084	0.0.0.0	255.255.255.255	UDP	64	Source port: 6808 Destination port: 6809
5	3.200143	0.0.0.0	255.255.255.255	UDP	64	Source port: 6808 Destination port: 6809
6	3.211128	0.0.0.0	255.255.255.255	UDP	64	Source port: 6808 Destination port: 6809
7	3.222119	0.0.0.0	255.255.255.255	UDP	64	Source port: 6808 Destination port: 6809
8	3.235337	0.0.0.0	255.255.255.255	UDP	64	Source port: 6808 Destination port: 6809
9	3.246123	0.0.0.0	255.255.255.255	UDP	64	Source port: 6808 Destination port: 6809
10	3.257120	0.0.0.0	255.255.255.255	UDP	64	Source port: 6808 Destination port: 6809
11	3.268141	0.0.0.0	255.255.255.255	UDP	64	Source port: 6808 Destination port: 6809
12	3.279135	0.0.0.0	255.255.255.255	UDP	64	Source port: 6808 Destination port: 6809
13	3.388272	0.0.0.0	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x726b6253
14	4.429253	0.0.0.0	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0x726b6253
15	5.449045	fe80::2d1:13ff:fe11:36ed	ff02::2	ICMPv6	70	Router Solicitation from 00:d1:13:11:36:ed
16	14.391238	00:d1:13:11:36:ed	Broadcast	ARP	64	Who has 192.168.52.1? Tell 192.168.52.14
17	81.483545	fe80::2d1:13ff:fe11:36ed	ff02::1:ff21:d362	ICMPv6	86	Neighbor Solicitation for fe80::2e39:96ff:fe2

```

> Frame 3: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
> Ethernet II, Src: 00:d1:13:11:36:ed (00:d1:13:11:36:ed), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 0.0.0.0 (0.0.0.0), Dst: 255.255.255.255 (255.255.255.255)
> User Datagram Protocol, Src Port: 6808 (6808), Dst Port: 6809 (6809)
▼ Data (18 bytes)
  Data: 48656c6c6f2c49276d206f6e206c696e6521
  [Length: 18]
0000  ff ff ff ff ff ff 00 d1 13 11 36 ed 08 00 45 00  .....6...E.
0010  00 2e 00 00 40 00 40 11 3a c0 00 00 00 00 ff ff  ....@.@. :.....
0020  ff ff 1a 98 1a 99 00 1a 91 d9 48 65 6c 6c 6f 2c  ....Hello,
0030  49 27 6d 20 6f 6e 20 6c 69 6e 65 21 00 00 00 00  I'm on line!....
    
```

On observe essentiellement :

- l'envoi de datagrammes UDP en broadcast IP contenant le message "Hello, I'm online!"
- une requête ARP de l'adresse IP ici 192.168.52.14 qui est donc celle de la caméra

Pour vérifier, on va se connecter au serveur DHCP puis rechercher l'adresse IP attribuée à partir de l'adresse MAC de la caméra :

```

$ ssh toto@192.168.52.83

# cat /var/lib/dhcp/dhcpd.leases | grep -A 1 -B 5 -i "00:D1:13:11:36:ED"
lease 192.168.52.216 {
    starts 3 2015/09/09 13:09:31;
    ends 3 2015/09/09 21:09:31;
    tstp 3 2015/09/09 21:09:31;
    binding state free;
    hardware ethernet 00:d1:13:11:36:ed;
    uid "\001\000\321\023\0216\355";
--
lease 192.168.52.216 {
    starts 5 2015/09/25 06:06:29;
    ends 5 2015/09/25 14:06:29;
    binding state active;
    next binding state free;
    hardware ethernet 00:d1:13:11:36:ed;
    uid "\001\000\321\023\0216\355";
    
```

SSH (Secure Shell) est à la fois un programme informatique et un protocole de communication sécurisé. Le protocole de connexion impose un échange de clés de chiffrement en début de connexion. Le protocole SSH a été conçu avec l'objectif de remplacer les différents programmes rlogin, telnet, rcp, ftp et rsh.

Maintenant, on va vérifier l'état de la communication IP avec la caméra :

```

$ ping 192.168.52.216 -c 1
PING 192.168.52.216 (192.168.52.216) 56(84) bytes of data.
    
```

```
64 bytes from 192.168.52.216: icmp_req=1 ttl=64 time=0.432 ms
```

```
--- 192.168.52.216 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.432/0.432/0.432/0.000 ms
```

ping est le nom d'une commande informatique réseau permettant d'envoyer une requête ICMP (demande d'ECHO ou echo-request) d'une machine à une autre machine (qui retourne une réponse d'ECHO ou echo-reply). Selon la réponse on connaît l'état de la machine distante. Si la machine ne répond pas il se peut que l'on ne puisse tout simplement pas communiquer avec elle. Cette commande réseau de base permet donc d'obtenir des informations et en particulier le temps de réponse de la machine à travers le réseau et aussi quel est l'état de la connexion avec cette machine (renvoi d'un code d'erreur correspondant).

## Services disponibles

La caméra IP fournit un certain nombre de services. Pour les découvrir, il suffit de “scanner” les ports ouverts sur la caméra. En effet, pour accéder à un service distant (notion de serveur offrant un service), il faut pouvoir l'identifier et cela est réalisé par le numéro de port. On peut remarquer que l'adresse IP ne permet que d'identifier la “machine” avec laquelle on communique mais cela ne suffira pas pour identifier l'application avec laquelle on veut communiquer.

Pour trouver les numéros de port “ouvert”, on va utiliser l'outil nmap côté client :

```
$ sudo apt-get install nmap

$ nmap -A -T4 192.168.52.216

Starting Nmap 5.21 ( http://nmap.org ) at 2015-09-25 08:28 CEST
Nmap scan report for 192.168.52.216
Host is up (0.015s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE      VERSION
23/tcp    open  telnet       BusyBox telnetd
99/tcp    open  http         GoAhead httpd (WAP http config)
| http-auth: HTTP Service requires authentication
|_ Auth type: Digest, realm = GoAhead
|_html-title: Document Error: Unauthorized
8600/tcp  open  tcpwrapped
Service Info: Device: WAP

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.40 seconds

$ nmap -p 21-23 192.168.52.216

Starting Nmap 5.21 ( http://nmap.org ) at 2015-09-25 08:34 CEST
Nmap scan report for 192.168.52.216
Host is up (0.00071s latency).
PORT      STATE SERVICE
21/tcp    closed ftp
22/tcp    closed ssh
23/tcp    open  telnet

Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds
```

On en conclut que :

- un serveur HTTP s'exécute sur le port 99 (et pas 80, le port par défaut pour les serveurs web)
- un serveur telnet s'exécute sur son port par défaut le 23

Telnet (TERminal NETwork ou TELEcommunication NETwork, ou encore TELetype NETwork) est un protocole utilisé sur tout réseau TCP/IP, permettant de communiquer avec un serveur distant en échangeant des lignes de textes et en recevant des réponses également sous forme de texte. Créé en 1969, telnet est un moyen de communication très généraliste et bi-directionnel. Il appartient à la couche application du modèle OSI et du modèle ARPA (DoD). Il est normalisé par l'IETF (RFC 15, 854 et 855). Il était notamment utilisé pour administrer des serveurs UNIX distant ou de l'équipement réseau, avant de tomber en désuétude par défaut de sécurisation, le texte étant échangé en clair, et l'adoption de SSH.

On peut vérifier le port affecté au service telnet dans la liste des ports *well-known* (bien connus) :

```
$ cat /etc/services | grep telnet
telnet      23/tcp
...
```

## Accès web : test en ligne de commande

Il est possible de tester l'accès aux CGI du serveur web de la caméra en ligne de commandes.

CGI (Common Gateway Interface) est une interface utilisée par les serveurs HTTP. Elle a été normalisée par la RFC 3875. Au lieu d'envoyer le contenu d'un fichier (fichier HTML, image), le serveur HTTP exécute un programme, puis retourne le contenu généré. CGI est le standard industriel qui indique comment transmettre la requête du serveur HTTP au programme, et comment récupérer la réponse générée. Une des caractéristiques de l'interface CGI est d'être indépendante de tout langage de programmation, car elle utilise les flux standard et les variables d'environnement. Même si le langage Perl a historiquement été souvent utilisé pour en écrire, il est possible d'écrire un programme CGI en C, Python, PHP, script shell, en VB ou en tout autre langage de programmation.

On va capturer le trafic en filtrant celui en provenance de l'adresse IP de la caméra :

**Wireshark: Capture Options**

**Capture**

Interface:  ▾

IP address: 192.168.52.2, fe80::9eb6:54ff:feb9:9222

Link-layer header type:  ▾

Buffer size:  ▾ megabyte(s)

Capture packets in promiscuous mode

Capture packets in monitor mode

Capture packets in pcap-ng format

Limit each packet to  ▾ bytes

Capture Filter:  ▾

**Capture File(s)**

File:

Use multiple files

Next file every  ▾ megabyte(s) ▾

Next file every  ▾ minute(s) ▾

Ring buffer with  ▾ files

Stop capture after  ▾ file(s)

**Stop Capture ...**

... after  ▾ packet(s)

... after  ▾ megabyte(s) ▾

... after  ▾ minute(s) ▾

**Display Options**

Update list of packets in real time

Automatic scrolling in live capture

Hide capture info dialog

**Name Resolution**

Enable MAC name resolution

Enable network name resolution

Enable transport name resolution

On démarre la capture puis on exécute une requête :

```
$ curl -v -X GET 'http://192.168.52.14:99/get_params.cgi?user=admin&pwd='
* About to connect() to 192.168.52.14 port 99 (#0)
* Trying 192.168.52.14... connected
> GET /get_params.cgi?user=admin&pwd= HTTP/1.1
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0
> Host: 192.168.52.14:99
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Mon Sep 28 16:43:42 2015
< Server: GoAhead-Webs
< Last-modified: Mon Mar 2 20:56:00 1970
< Content-type: text/html
< Cache-Control:no-cache
```



```
< Content-length: 3331
< Connection: close
<
var now1=1443458622;
var tz=-28800;
var ntp_enable=1;
var ntp_svr="time.nist.gov";
var dhcpen=1;
var ip="192.168.52.14";
var mask="255.255.255.0";
...
```

Le protocole NTP (Network Time Protocol) est un protocole qui permet de synchroniser, via un réseau informatique, l'horloge locale d'un ordinateur sur une référence d'heure.

On a capturé un échange TCP :

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	9c:b6:54:b9:92:22	ff:ff:ff:ff:ff:ff	ARP	42	Who has 192.168.52.14? Tell 192.168.52.2
2	0.000215	00:d1:13:11:36:ed	9c:b6:54:b9:92:22	ARP	64	192.168.52.14 is at 00:d1:13:11:36:ed
3	0.000222	192.168.52.2	192.168.52.14	TCP	74	46658 > metagram [SYN] Seq=0 Win=14600 Len=0
4	0.000659	192.168.52.14	192.168.52.2	TCP	74	metagram > 46658 [SYN, ACK] Seq=0 Ack=1 Win=5
5	0.000675	192.168.52.2	192.168.52.14	TCP	66	46658 > metagram [ACK] Seq=1 Ack=1 Win=14720
6	0.000718	192.168.52.2	192.168.52.14	TCP	264	46658 > metagram [PSH, ACK] Seq=1 Ack=1 Win=14
7	0.001153	192.168.52.14	192.168.52.2	TCP	66	metagram > 46658 [ACK] Seq=1 Ack=199 Win=6864
8	0.008318	192.168.52.14	192.168.52.2	TCP	115	metagram > 46658 [PSH, ACK] Seq=1 Ack=199 Win=
9	0.008324	192.168.52.2	192.168.52.14	TCP	66	46658 > metagram [ACK] Seq=199 Ack=50 Win=147
10	0.008672	192.168.52.14	192.168.52.2	TCP	88	metagram > 46658 [PSH, ACK] Seq=50 Ack=199 Win
11	0.008677	192.168.52.2	192.168.52.14	TCP	66	46658 > metagram [ACK] Seq=199 Ack=72 Win=147
12	0.009174	192.168.52.14	192.168.52.2	TCP	107	metagram > 46658 [PSH, ACK] Seq=72 Ack=199 Win
13	0.009179	192.168.52.2	192.168.52.14	TCP	66	46658 > metagram [ACK] Seq=199 Ack=113 Win=14
14	0.013438	192.168.52.14	192.168.52.2	TCP	91	metagram > 46658 [PSH, ACK] Seq=113 Ack=199 W
15	0.013447	192.168.52.2	192.168.52.14	TCP	66	46658 > metagram [ACK] Seq=199 Ack=138 Win=14
16	0.013774	192.168.52.14	192.168.52.2	TCP	90	metagram > 46658 [PSH, ACK] Seq=138 Ack=199 W
17	0.013780	192.168.52.2	192.168.52.14	TCP	66	46658 > metagram [ACK] Seq=199 Ack=162 Win=14

▶ Frame 3: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)  
 ▶ Ethernet II, Src: 9c:b6:54:b9:92:22 (9c:b6:54:b9:92:22), Dst: 00:d1:13:11:36:ed (00:d1:13:11:36:ed)  
 ▶ Internet Protocol Version 4, Src: 192.168.52.2 (192.168.52.2), Dst: 192.168.52.14 (192.168.52.14)  
 ▶ Transmission Control Protocol, Src Port: 46658 (46658), Dst Port: metagram (99), Seq: 0, Len: 0

```
0000  00 d1 13 11 36 ed 9c b6 54 b9 92 22 08 00 45 00  ....6... T.."..E.
0010  00 3c d0 d0 40 00 40 06 80 8a c0 a8 34 02 c0 a8  .<..@.@. ....4...
0020  34 0e b6 42 00 63 f1 6f e4 ff 00 00 00 00 a0 02  4..B.C.o .....
0030  39 08 58 15 00 00 02 04 05 b4 04 02 08 0a 0c 3d  9.X..... ==
0040  34 2f 00 00 00 00 01 03 03 07 4/..... ..
```

Ce qui est intéressant avec Wireshark, c'est qu'il lui est possible de reconstituer l'échange en sélectionnant une trame puis en choisissant **Follow TCP Stream** à partir du menu accessible avec le bouton droit de la souris :



1	0.000000	9c:b6:54:b9:92:22	ff:ff:ff:ff:ff:ff	ARP	42	Who has 192.168.52.14? Tell 192.168.52.2
2	0.000215	00:d1:13:11:36:ed	9c:b6:54:b9:92:22	ARP	64	192.168.52.14 is at 00:d1:13:11:36:ed
3	0.000222	192.168.52.2	192.168.52.14	TCP	74	46658 > metagram >
4	0.000659	192.168.52.14	192.168.52.2	TCP	74	metagram >
5	0.000675	192.168.52.2	192.168.52.14	TCP	66	46658 > metagram >
6	0.000718	192.168.52.2	192.168.52.14	TCP	264	46658 > metagram >
7	0.001153	192.168.52.14	192.168.52.2	TCP	66	metagram >
8	0.008318	192.168.52.14	192.168.52.2	TCP	115	metagram >
9	0.008324	192.168.52.2	192.168.52.14	TCP	66	46658 > metagram >
10	0.008672	192.168.52.14	192.168.52.2	TCP	88	metagram >
11	0.008677	192.168.52.2	192.168.52.14	TCP	66	46658 > metagram >
12	0.009174	192.168.52.14	192.168.52.2	TCP	107	metagram >
13	0.009179	192.168.52.2	192.168.52.14	TCP	66	46658 > metagram >
14	0.013438	192.168.52.14	192.168.52.2	TCP	91	metagram >
15	0.013447	192.168.52.2	192.168.52.14	TCP	66	46658 > metagram >
16	0.013774	192.168.52.14	192.168.52.2	TCP	90	metagram >
17	0.013780	192.168.52.2	192.168.52.14	TCP	66	46658 > metagram >

▸ Frame 3: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)  
 ▸ Ethernet II, Src: 9c:b6:54:b9:92:22 (9c:b6:54:b9:92:22), Dst: 00:d1:13:11:36:ed (00:d1:13:11:36:ed)  
 ▸ Internet Protocol Version 4, Src: 192.168.52.2 (192.168.52.2), Dst: 192.168.52.14 (192.168.52.14)  
 ▸ Transmission Control Protocol, Src Port: 46658 (46658), Dst Port: metagram (99), Seq: 0, Len: 74

```

0000  00 d1 13 11 36 ed 9c b6 54 b9 92 22 08 00 45 00  ....6... T..."..E.
0010  00 3c d0 d0 40 00 40 06 80 8a c0 a8 34 02 c0 a8  .<..@.@. ....4...
0020  34 0e b6 42 00 63 f1 6f e4 ff 00 00 00 00 a0 02  4..B.c.o .....
0030  39 08 58 15 00 00 02 04 05 b4 04 02 08 0a 0c 3d  9.X..... =
0040  34 2f 00 00 00 00 01 03 03 07                    4/..... ..
  
```

- Mark Packet (toggle)
- Ignore Packet (toggle)
- Set Time Reference (toggle)
- Manually Resolve Address
- Apply as Filter >
- Prepare a Filter >
- Conversation Filter >
- Colorize Conversation >
- SCTP >
- Follow TCP Stream
- Follow UDP Stream
- Follow SSL Stream
- Copy >
- Decode As...
- Print...
- Show Packet in New Window

On obtient :

**Follow TCP Stream**

**Stream Content**

```
GET /get_params.cgi?user=admin&pwd= HTTP/1.1
User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4
libidn/1.23 librtmp/2.3
Host: 192.168.52.14:99
Accept: */*

HTTP/1.1 200 OK
Date: Mon Dec 7 16:43:06 2015
Server: GoAhead-Webs
Last-modified: Mon Mar 2 20:56:00 1970
Content-type: text/html
Cache-Control:no-cache
Content-length: 3331
Connection: close

var now1=1449506585;
var tz=-28800;
var ntp_enable=1;
var ntp_svr="time.nist.gov";
var dhcp=1;
var ip="192.168.52.14";
var mask="255.255.255.0";
var gateway="192.168.52.1";
var dns1="8.8.8.8";
var dns2="192.168.52.1";
var port=99;
var dev2_host="192.168.1.111";
```

Entire conversation (3733 bytes)

Rechercher Enregistrer sous Imprimer  ASCII  EBCDIC  Hex Dump  C Arrays  Raw

Aide Filter Out This Stream Fermer

## Accès telnet

Pour accéder au serveur telnet de la caméra, il faut un compte. Ici, il apparaît que le compte `root` n'est pas le même que celui pour administrer la caméra à partir de l'interface web (port 99). Obtenir le mot de pass de `root` sort du cadre de ce document. Mais on peut décrire la procédure si le système Linux est l'OS de la caméra :

- obtenir le *firmware* par le site du fabricant (c'est une habitude des constructeurs de le fournir car cela permet de mettre à jour le "logiciel" embarqué dans la caméra)
- analyser le *firmware* (un outil comme *binwalk* peut s'avérer utile)
- rechercher le fichier contenant le compte `root` et son mot de passe crypté (généralement un fichier `passwd` et éventuellement le fichier `shadow`)
- casser le mot de passe avec un logiciel comme *John the Ripper* (un outil comme *hydra* peut aussi être utilisé)

Pour ce type de caméra IP, le *firmware* est un fichier portant l'extension `.bin` :

```
// on recherche l'identifiant (PK) des fichiers ZIP
$ hexdump -n 160 -C 67.2.2.172.bin
00000000  77 69 66 69 2d 63 61 6d 65 72 61 2d 73 79 73 2d |wifi-camera-sys-|
00000010  71 65 74 79 69 70 61 64 67 6a 6c 7a 63 62 6d 6e |qetyipadgjlzcbmn|
00000020  25 0d 05 00 50 4b 03 04 0a 00 00 00 00 00 c2 99 |%...PK.....|
00000030  6c 43 00 00 00 00 00 00 00 00 00 00 00 07 00 |lC.....|
00000040  1c 00 73 79 73 74 65 6d 2f 55 54 09 00 03 7b 0d |..system/UT...{|
```

```
00000050 82 52 3d 12 7b 53 75 78 0b 00 01 04 fe ff 00 00 |.R={Sux.....|
00000060 04 fe ff 00 00 50 4b 03 04 0a 00 00 00 00 c2 |....PK.....|
00000070 99 6c 43 00 00 00 00 00 00 00 00 00 00 0b |.LC.....|
00000080 00 1c 00 73 79 73 74 65 6d 2f 77 77 77 2f 55 54 |...system/www/UT|
00000090 09 00 03 7b 0d 82 52 3d 12 7b 53 75 78 0b 00 01 |...{.R={Sux...|
000000a0
```

```
// on renomme alors le fichier en .zip
```

```
$ mv 67.2.2.172.bin 67.2.2.172.zip
```

```
// on décompresse l'archive
```

```
$ unzip 67.2.2.172.zip
```

```
Archive: 67.2.2.172.zip
```

```
warning [67.2.2.172.zip]: 36 extra bytes at beginning or within zipfile
(attempting to process anyway)
```

```
creating: system/
```

```
creating: system/www/
```

```
creating: system/system/
```

```
creating: system/system/bin/
```

```
inflating: system/system/bin/daemon.v5.9
```

```
inflating: system/system/bin/encoder
```

```
inflating: system/system/bin/gmail_thread
```

```
creating: system/system/lib/
```

```
creating: system/system/drivers/
```

```
creating: system/Wireless/
```

```
creating: system/init/
```

```
inflating: system/init/ipcam.sh
```

```
// on recherche le fichier passwd (et éventuellement shadow)
```

```
$ strings system/system/bin/daemon.v5.9 | grep -A 1 -B 1 passwd
/system/system/bin/encoder &
/etc/passwd
```

```
root:LSiuY7p0mZG2s:0:0:Adminstrator:/:/bin/sh
```

```
--
```

```
recv failed by zqh errno=%d
```

```
user or passwd is error
```

```
rm -f /tmp/post1.bin
```

```
check user or passwd is ok
```

```
GCC: (GNU) 3.3.2
```

```
// on peut rechercher si il y a des décompressions (avec unzip) avec mot de passe
```

```
$ strings system/system/bin/daemon.v5.9 | grep -A 1 -B 1 unzip
```

```
cp /tmp/system-b.ini /system/www/system-b.ini
```

```
unzip1 -o -P vstarcam!@#$$% /tmp/www.zip -d /system
```

```
rm /tmp/www.zip
```

```
--
```

```
First Extract all file Except daemon
```

```
unzip -o /tmp/system.zip -x system/system/bin/daemon* -d /.
```

```
Sencond Extract daemon only
```

```
unzip -o /tmp/system.zip system/system/bin/daemon* -d /.
```

```
chmod a+x /system/system/bin/*
```

```
// il faut donc décrypter la chaîne LSiuY7p0mZG2s (cf. John the Ripper)
```

```
// on obtient le mot de passe "123456"
```

```
// vérifions le mot de passe obtenu
```

```
$ perl -e "print crypt('123456','LS');"
```

```
LSiuY7p0mZG2s
```

John the Ripper (ou JTR, ou John) est un logiciel libre de cassage de mot de passe, utilisé notamment pour tester la sécurité d'un mot de passe (audit, crack). D'abord développé pour tourner sous les systèmes dérivés d'UNIX, le programme fonctionne aujourd'hui sous une cinquantaine de plates-formes différentes, telles que BeOS, BSD et ses dérivés, DOS, Linux, OpenVMS, Win32 ... John est l'un des craqueurs de mots de passe les plus populaires, car il inclut l'autodétection des tables de hachage utilisées par les mots de passe, l'implémentation d'un grand nombre d'algorithmes de cassage, par le fait qu'il soit très facilement modifiable, et aussi qu'il soit possible de reprendre une attaque après une pause (arrêt de la machine).

Ici, le mot de passe par défaut du compte root est 123456 !

```
$ telnet 192.168.52.216
Trying 192.168.52.216...
Connected to 192.168.52.216.
Escape character is '^]'.

(none) login: root
Password: 123456

BusyBox v1.12.1 (2012-11-20 15:16:24 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

# ls
var      usr      tmp      system  sys      sbin     proc     param   mnt     media   lib
init     home    etc_ro  etc     dev     bin

# ifconfig
eth2      Link encap:Ethernet  HWaddr 00:D1:13:11:36:ED
...
```

BusyBox est un logiciel qui implémente un grand nombre des commandes standard sous Unix, à l'instar des GNU Core Utilities. Comme chaque fichier binaire exécutable pour Linux comporte plusieurs kilooctets d'informations additionnelles, l'idée de combiner plus de deux cent programmes en un seul fichier exécutable permet de gagner une taille considérable. Distribué sous la licence GNU GPL version 2, BusyBox est un logiciel libre.

Pour connaître l'architecture processeur de la caméra IP, on peut faire :

```
# cat proc/cpuinfo
system type      : Ralink SoC
processor        : 0
cpu model       : MIPS 24K V4.12
BogoMIPS        : 239.10
...

# free
              total        used         free       shared    buffers
Mem:          29336         21940         7396           0         1012
Swap:           0           0           0
Total:        29336         21940         7396
```

Et pour son système d'exploitation :

```
# cat proc/version
Linux version 2.6.21 (root@mailzxh-desktop) (gcc version 3.4.2) #653 Tue Nov 20 15:22:24 CST 2012
```

Sur les systèmes du type Unix/Linux, procfs (process file system, système de fichiers processus) est un pseudo-système de fichiers (pseudo car dynamiquement généré au démarrage) utilisé pour accéder aux informations du noyau sur les processus. Le système de fichiers est souvent monté sur le répertoire `/proc`. Puisque `/proc` n'est pas une arborescence réelle, il ne consomme aucun espace disque mais seulement une quantité limitée de mémoire vive. Une modification d'un fichier situé dans `/proc` ne sera donc pas permanente. Certains fichiers ne sont accessibles qu'en lecture seule.

Conclusion : la caméra est équipée d'un processeur Ralink MIPS CPU à 360Mhz, avec 32MB de RAM, et d'un système d'exploitation Linux 2.6.21.

On termine notre petit tour par une observation des serveurs actifs :

```
# netstat -nap
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp      0      0 0.0.0.0:99             0.0.0.0:*                LISTEN      147/encoder
```

tcp	0	0 0.0.0.0:23	0.0.0.0:*	LISTEN	28/telnetd
tcp	0	0 0.0.0.0:8600	0.0.0.0:*	LISTEN	29/daemon.v5.5
tcp	0	0 192.168.52.14:23	192.168.52.2:43586	ESTABLISHED	28/telnetd
udp	472	0 127.0.0.1:8832	0.0.0.0:*		147/encoder
udp	0	0 127.0.0.1:6666	0.0.0.0:*		147/encoder
udp	0	0 0.0.0.0:8600	0.0.0.0:*		29/daemon.v5.5
udp	0	0 127.0.0.1:9123	0.0.0.0:*		29/daemon.v5.5
udp	0	0 127.0.0.1:9124	0.0.0.0:*		147/encoder
udp	0	0 0.0.0.0:32108	0.0.0.0:*		147/encoder
udp	0	0 127.0.0.1:8812	0.0.0.0:*		31/gmail_thread
udp	0	0 127.0.0.1:8813	0.0.0.0:*		147/encoder
udp	0	0 127.0.0.1:8822	0.0.0.0:*		147/encoder
udp	0	0 0.0.0.0:20989	0.0.0.0:*		147/encoder
udp	0	0 127.0.0.1:8831	0.0.0.0:*		30/cmd_thread

On en déduit que :

- le serveur web (HTTP) est le processus `encoder` (!) sur le port 99
- le serveur telnet est `telnetd` sur le port 23
- et qu'il y a un serveur qui s'exécute sur le port 8600 (?)

Cela correspond évidemment à ce que l'on avait obtenu avec `nmap` côté client.

Lorsque le système d'exploitation de la caméra démarre à la mise sous tension, celui-ci exécutera un script qui lance les serveurs identifiés précédemment :

```
// Que se passe-t-il au démarrage ?
# cat etc_ro/inittab
::sysinit:/etc_ro/rcS
ttyS1::respawn:/bin/sh

// on lance le script rc suivant :
# cat etc_ro/rcS
#!/bin/sh
mount -a
mkdir -p /var/run
cat /etc_ro/motd
nvram_daemon&
#goahead&
#for telnet debugging
#telnetd
#for syslogd
#/usr/sbin/network.sh
mkdir -p /var/log
mount -t jffs2 /dev/mtdblock6 /system
mount -t jffs2 /dev/mtdblock7 /param
/system/init/ipcam.sh

// qui se termine par l'exécution du script :
# cat system/init/ipcam.sh
export LD_LIBRARY_PATH=/system/system/lib:$LD_LIBRARY_PATH
export PATH=/system/system/bin:$PATH
telnetd
/system/system/bin/daemon.v5.5 &
/system/system/bin/cmd_thread &
/system/system/bin/gmail_thread &
```

Ne surtout pas modifier le mot de passe de root et le script `ipcam.sh`.

Le contrôle d'accès des pages web est réalisé à partir du fichier `login.cgi` :

```
# cat param/login.cgi
var loginuser="admin";
var loginpass="";
var pri=255;
```

Mais ce fichier est ré-écrit par le processus `encoder` (le serveur web) à partir du fichier `system.ini` qui contient la configuration réalisée par l'interface web. Ce fichier est un format binaire propre à l'application `encoder` contenant des chaînes des caractères dont le compte et mot de passe de l'administrateur.

On va afficher le contenu du fichier `system.ini` :

```
# cd /system/www/

# vi lire.sh
#!/bin/sh

# affiche un caractere visible
chr(){
    n=`expr "$1" : '[:print:]*'`
    if [ "$n" -eq 0 ]
    then
        # on remplace par le caractere %
        echo -n '%'
        return 1
    fi
    echo -n $1
}

# enleve les caracteres de controle
#cat "$1" | sed -e 's/[:cntrl:]/ /g' > ".$1"
cat "$1" > ".$1"

# affiche le fichier comme la commande strings
for ligne in $(cat ".$1");
do
    len=${#ligne}
    let i=0
    while [ $i -lt $len ]
    do
        a=${ligne:$i:1}
        chr $a
        let i=i+1
    done
    echo ""
done

rm -f ".$1"

# ./lire.sh system.ini
JWEV-182544-CFCFFIPCAM%%C00:D1:13:11:36:ED%%00:D1:13:11:36:EE%%yHV%%time.nist.gov
http://ipcnp.com/upgengxin.aspP%P%192.168.52.216255.255.255.0192.168.52.18.8.8.8192.168.52.1%c
%admin%wanscam_office88888888%PIPCAM0123456789192.168.9.1255.255.255.0192.168.9.2192.168.9.254
%%<%%%%@P%%
%@%192.168.1.111IPCAMipcc
```

Ici, le compte administrateur de l'interface web est `admin` sans mot de passe.

Les fichiers composant l'interface web sont stockés à la racine du serveur HTTP :

```
# ls -l system/www/
system/www/Japanese/          system/www/codebase/          system/www/italian/
system/www/public.js          system/www/spanish/           system/www/Korean/
system/www/config.htm         system/www/jquery-1.3.1.js    system/www/reboot.htm
system/www/status.htm         system/www/Polski/           system/www/control.htm
system/www/jquery/           system/www/rebootme.htm       system/www/style.css
system/www/Portugal/         system/www/datetime.htm       system/www/live.htm
system/www/recordpath.htm     system/www/swedish/          system/www/Russian/
system/www/ddns.htm           system/www/log.htm            system/www/recordpath2.htm
system/www/system-b.ini       system/www/admin.htm          system/www/decoder.htm
system/www/login.htm          system/www/recordplay.htm     system/www/system.ini
...
```

Pour éditer un fichier, on peut évidemment utiliser le célèbre vi (voilà à quoi il sert!) :

```
# vi system/www/index.htm :
```

On constate qu'il existe une variable `language` qui permet de sélectionner la langue par défaut pour l'interface web (la valeur 3 correspond au français).

Pour sortir d'une session telnet, il suffit de faire :

```
# exit
```

Il est évidemment possible de *sniffer* le trafic de la communication `telnet` pour vérifier si l'authentification (*login/password*) est transmise en clair ou sous une forme cryptée.

Pour cela, vous pouvez utiliser **Wireshark** (en appliquant la règle de filtrage `tcp.port == 23`) :

```
$ gksudo wireshark
```

Ou `tcpdump` en ligne de commandes :

```
$ sudo tcpdump -Av port 23  
...
```

```
Conclure sur le niveau de sécurité de ce type de caméra.
```

[Retour au sommaire](#)