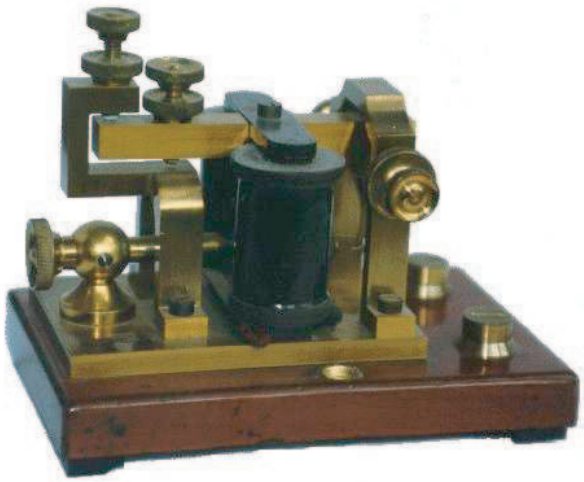
Mécanismes de configuration automatique d'une interface réseau, aspects sécurité

Brian Amedro Vladimir Bodnartchouk Vincent Robitzer



Université Nice SOPHIA ANTIPOLIS

Mécanismes de configuration automatique d'une interface réseau, aspects sécurité

AMEDRO Brian BODNARTCHOUK Vladimir ROBITZER Vincent

27 juin 2005

Préface

Aujourd'hui, les réseaux constituent un domaine privilégié qui fournit à l'homme des services irremplaçables.

Cepandant, les mécanismes de configuration qui permettent à un ordinateur d'exister au sein d'un réseau, constituent une étape délicate, facilement exploitable par des utilisateurs malveillants, comme Brian en a fait l'expérience lors de l'organisation de tournois de jeux vidéo faisant parfois intervenir plus de 300 machines.

Il devient donc indispensable de compter sur des mécanismes entièrement automatisés et securisés.

Brian AMEDRO
Vladimir BODNARTCHOUK
Vincent ROBITZER

Table des matières

1 Enjeux de la configuration d'une interface réseau **4**

I Protocoles existants 6

2 Chronologie **7**

3 RARP **9**

4 BootP **10**

5 AutoIP **12**

6 DHCP **14**
6.1 Principe de fonctionnement . . . 15
6.1.1 Le serveur DHCP . . . 15
6.1.2 Le client DHCP . . . 15
6.1.3 L'agent relais . . . 16
6.2 Exemple de connexion . . . 16
6.3 Les paquets . . . 16
6.4 Les avantages de ce protocole . . . 17
6.5 Les inconvénients de ce protocole . . . 17
6.5.1 Problèmes divers . . . 17
6.5.2 Problèmes liés à la sécurité . . . 17

II Proposition d'un protocole alternatif 19

7 Le protocole SDHCP **21**
7.1 Les choix techniques . . . 22
7.1.1 RSA . . . 22
7.1.2 SHA . . . 22
7.2 Format d'un paquet SDHCP . . . 23
7.3 Diagramme de séquence . . . 24
7.4 Les messages SDHCP . . . 25

8 Les attaques **26**
8.1 Le rejeu . . . 27
8.2 Le passeur de seau . . . 28
8.3 Le faux serveur SDHCP . . . 29

9 Implémentation d'un prototype **30**
9.1 Fonctionnement de l'application . . . 31
9.2 Etude des choix techniques . . . 32
9.2.1 La librairie OpenSSL . . . 32
9.3 Conclusion sur ce prototype . . . 33

Bibliographie **34**

Introduction

Pour qu'un ordinateur puisse exister au sein d'un réseau, il faut le configurer. Cette opération délicate peut être effectuée soit par un administrateur, soit de façon automatique.

L'objectif de cette étude est, après avoir définit plus en détails les enjeux de la configuration d'une machine, d'analyser dans une première partie les différents mécanismes permettant d'automatiser cette opération. Après une description des protocoles AutoIP et BootP, une attention particulière sera portée sur le protocole DHCP[1] qui est le mécanisme le plus répandu. Un examen attentif de son fonctionnement sera donc effectué, et nous verrons ses avantages et inconvénients, notamment concernant la sécurité.

Une seconde partie sera ensuite consacrée à la proposition d'un protocole alternatif sécurisé, basé sur l'authentification des clients. Nous verrons comment celui-çi résoud certains problèmes de DHCP. Nous soumettrons alors une implémentation de ce protocole ainsi que des tests de ce dernier.

[1] DHCP : Dynamic Host Configuration Protocol

Chapitre 1

Enjeux de la configuration d'une interface réseau

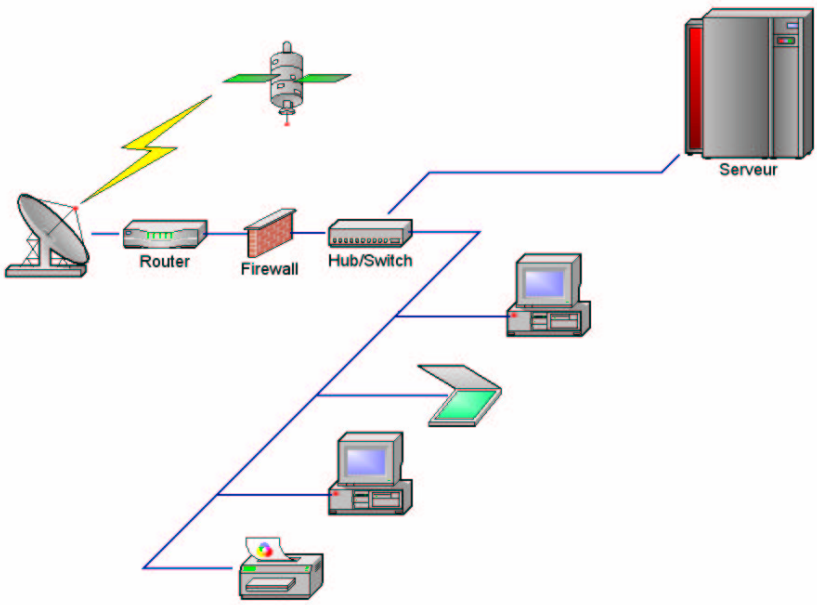
Dans un réseau TCP/IP, chaque carte réseau est identifiée par deux adresses. Une adresse physique, et une adresse logique.

L'adresse physique, appelée aussi adresse MAC[1], est un nombre de 128 bits attribué à la carte réseau lors de sa fabrication. Ce nombre est unique et il n'existe donc pas deux cartes ayant une même adresse physique.

L'adresse logique, appelée aussi adresse IP[2], est un nombre de 32 bits permettant d'identifier de façon unique une machine sur un réseau. Cette adresse peut ainsi inclure :

- un numéro d'identification du réseau
- un numéro d'identification du sous-réseau
- un numéro d'hôte, identifiant la machine sur le réseau

Cette adresse IP donne donc la possibilité à l'administrateur de segmenter son réseau afin d'en optimiser le traffic.



[1]MAC : Media Access Control
[2]IP : Internet Protocol

Sur ce réseau, en plus des ordinateurs des utilisateurs, il peut aussi exister d'autres matériels comme un serveur DNS[3], une passerelle permettant d'accéder à internet, un serveur FTP[4], etc... Chacun de ces matériels possèdent eux aussi leur propre adresse IP.

Pour fonctionner correctement, chaque utilisateur devra donc avoir une machine configurée pour utiliser toutes les possibilités offertes par son réseau. Il faudra ainsi lui affecter, entre autres, une adresse IP, l'adresse de la passerelle internet, les adresses des serveurs DNS, etc... Il peut ainsi y avoir, pour certains réseaux, plus de 20 paramètres particuliers à définir.

Cette opération de configuration ne peut donc en général être effectuée par l'utilisateur car elle nécéssite à la fois des compétences techniques et une connaissance approfondie de l'architecture globale du réseau.

Dans une structure de quelques machines, tout ceci peut-être fait par l'administrateur lui-même, mais dès que le nombre de machines augmente, il devient très vite utile d'utiliser des outils qui permettront d'automatiser cette tâche. On imagine mal en effet l'administrateur intervenir dès qu'un utilisateur arrive avec son ordinateur portable.

[3]DNS : Domain Name Server
[4]FTP : File Transfert Protocol

Première partie

Protocoles existants

Chapitre 2

Chronologie

- 1969 :

  réseau expérimental ARPANET créé par l'ARPA reliant quatre ordinateurs de quatre universités différentes (Le Stanford Institute, L'université de Californie à Los Angeles, L'université de Californie à Santa Barbara, L'université d'Utah).

  Steve Crocker met au point le système RFC (Request For Comment). Il s'agit de documents présentés sous la forme de note permettant aux chercheurs d'échanger leurs travaux.

- 1970 :

  NPC premier protocole de communication créé par Steve Crocker, l'ancêtre de TCP/IP.

- 1971 :

  FTP protocole de transfert de donnée élaboré par A. Bhushan [RFC-114]

- 1972 :

  apparition du courier électronique par Ray Tomlinson, ARPANET est présenté au grand public.

- 1974 :

  V. Cerf et B. Kahn publient un article sur le protocole TCP.

- 1976 :

  protocole TCP déployé sur le réseau Arpanet (111 machines).

- 1978 :

  RSA (Rivest Shamir Adleman), algorithme cryptographique.

- 1981 :

apparition du protocole TFTP (Trivial File Transfert Protocol)

– 1983 :

division ARPANET : MILNET (réseau uniquemenet dedié à la transmission de données militaires) et ARPANET (dedié à la communauté scientifique).

DNS (Domain Name System) première spécification et implémentation par P. Mockapetris (ISI) [RFC-883]

– 1984 :

l'organisme de normalisation ISO (International Organization for Standardization), basé en Suisse, a développé un modèle de référence pour que les réseaux puissent se développer à l'échelle mondiale en dehors du cercle fermé de certaines entreprises et institutions.

apparition du protocole RARP (Reverse Address Resolution Protocol) crée par Finlayson, Mann, Mogul, Theimer[RFC-903]

– 1985 :

BOOTP protocole crée par Bill Croft (Université de Stanford) et John Gilmore (Sun Microsystems) [RFC-951].

– 1990 :

apparition du protocole HTTP, et du langage HTML par Tim Berners-Lee.

– 1992 :

MD5 (Message Digest 5), algorithme de signature inventé par Ronald Rivest.

– 1995 :

Internet, réseau public commercial.

spécification de IPv6 (Internet Protocol verion 6) par S. Deering (Xerox Parc) et R. Hinden (Ipsilon Networks) [RFC-1883].

SHA (Secure Hash Algorithm) est une fonction de hachage cryptographique conçue par la National Security Agency.

– 1997 :

apparition du protocole DHCP par l'auteur K. Droms (Université de Bucknell) [RFC-2131].

– 2003 :

DHCPv6 (DHCP pou IPv6) [RFC-3315].

Chapitre 3

RARP

Le protocole RARP (Reverse Address Resolution Protocol) est historiquement le premier mécanisme automatisé permettant l'assignation d'adresses IP.

Le fonctionnement est assez simple et facile à implementer, lorsque l'ordinateur est connecté au réseau il diffuse un message contentant son adresse MAC (identifiant physique unique), de son côté le serveur contient une table dans laquelle pour chaque adresse physique il y a une et une seule adresse IP. Ainsi lorsque le serveur reçoit un datagrame RARP de la part du client il repond avec l'adresse IP associée.

<table>
<tr><td>00</td><td>01</td><td>02</td><td>03</td><td>04</td><td>05</td><td>06</td><td>07</td><td>08</td><td>09</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td colspan="16">Type de matériel</td></tr>
<tr><td colspan="8">Longueur adresse matériel</td><td colspan="8">Longueur adresse protocole</td></tr>
<tr><td colspan="16">Type de logiciel</td></tr>
<tr><td colspan="8">Type de protocole</td><td colspan="8">Opcode</td></tr>
<tr><td colspan="16">Adresse MAC source</td></tr>
<tr><td colspan="16">Adresse IP source</td></tr>
<tr><td colspan="16">Adresse MAC destination</td></tr>
<tr><td colspan="16">Adresse IP destination</td></tr>
</table>

Fig. 3.1 – Exemple de paquet RARP

L'inconvénient majeur de ce protocole est le fait que le serveur renvoi uniquement l'adresse IP mais pas l'adresse de la passerelle, ni le masque de sous-réseau qui doivent être configurés d'une autre manière.
Un autre inconvenient concerne la cohabitation de plusieurs serveurs RARP dans le même réseau, lorsque plusieurs serveurs recoivent une demande d'adresse le client ne s'attends pas à ce que tous ces serveurs lui répondent ou lui répondent avec des adresses differentes.

Cependant il existe une variante dynamique de ce protocole comme le DRARP, créé pas Sun Microsystems en 1988 il rends homogène la communication entre plusieurs serveurs grâce à des mécanismes de synchronisation et de contrôle des réponses.

Chapitre 4

BootP

Le protocole BootP (Bootstrap Protocol) qui est une évolution directe de RARP a été initialement conçu pour activer la configuration d'amorçage des stations de travail sans disques durs.

Ce protocole, qui reprends l'idée de la centralisation des données de RARP, fait intervenir un serveur qui gère une liste statique de clients (identifiés par leurs adresses physiques) associées aux paramètres de configuration qui sont :

- adresse IP
- adresse du serveur et nom du fichier de démarrage
- adresse de la passerelle
- masque de sous-réseau

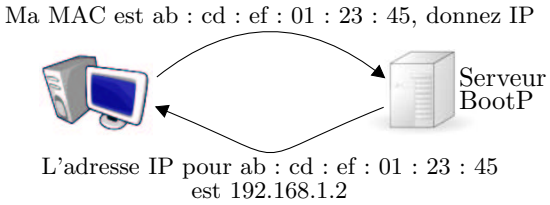


FIG. 4.1 – Schéma de fonctionnement du protocole BootP

Désormais les réponses envoyés par le serveur au client se font en unicast (datagramme UDP avec une destination précise et non en diffusion) ce qui n'était pas standardisé dans le protocole parent RARP. Parmi les évolutions notables nous pouvons remarquer le fait que que ce protocole peut être mis en place dans de grands réseaux grâce à l'Agent Relais BootP supporté par les routeurs (qui laissent passer sous certaines conditions des paquets de type BootP entre plusieurs réseaux inter-connectés). A noter que ce protocole permet de préciser des informations spécifiques du fournisseur de matériel grâce à un champs de 64 octets prévu dans le datagramme BootP [RFC-1084], typiquement afin de définir la configuration du routeur proxy.
Néanmoins, ce protocole oblige l'administrateur à attribuer une adresse IP fixe à chaque machine qui peut se connecter au réseau. Il en résulte un gaspillage d'adresses car une station qui se connecte rarement se voit attribuer une adresse malgré tout.

Pour éviter ce gaspillage le protocole DHCP qui est une évolution de BootP est apparu en 1997, il inclu un mécanisme de crédit-bail qui permet de contrôler le cycle de vie des adresses IP. (sera vu par la suite en détails)

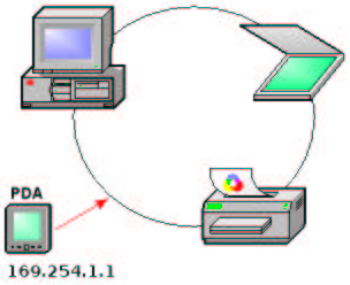
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
Opcode								Type de matériel							
Longueur adresse matériel								Compteur de sauts[1]							
ID Transaction															
Nombre de secondes															
Flags															
Adresse IP client															
Votre adresse IP															
Adresse IP serveur															
Adresse IP passerelle															
Adresse MAC client (16 octets)															
Nom du serveur (64 octets															
Nom du fichier d'amorçage (128 octets)															
Informations du fournisseur (64 octets)															

FIG. 4.2 – Exemple de paquet BootP

Chapitre 5

AutoIP

Les extensions du célèbre protocole DHCP décrivent de nouvelles méthodes d'attribution d'adresses IP sans l'intervention d'un administrateur et même en l'abscence d'un serveur.



ces mécanismes sont connus comme :
- Autonet
- Auto-IP Configuration
- IP-Auto Configuration
- Automatic Private IP Addressing (APIPA) par Microsoft

Ce n'est donc pas un protocole standardisé, au sens qu'il n'est pas présent dans la RFC. Pour cela nous l'appellerons plus simplement AutoIP.

Le fonctionnement d'AutoIP est assez simple, quand le client se connecte au réseau il se comporte tout d'abord comme un client DHCP puis si le serveur n'est pas disponible AutoIP séléctionne une adresse aléatoire dans l'intervalle 169.254.0.1 à 169.254.255.254. L'IANA (Internet Assigned Numbers Autority) a réservé cet intervalle d'adresses pour un usage privé, donc personne ne pourra l'utiliser sur Internet.
De plus le client utilisera son adresse IP autoconfiguré jusqu'à ce que le serveur DHCP devienne disponible (test de présence toutes les 3 minutes) et s'attribura par defaut le masque de sous-réseau 255.255.0.0.

Il est important de préciser que pour résoudre des conflits potentiels de duplication d'adresses IP, AutoIP utilise le protocole ARP (Address Resolution Protocol). Après avoir séléctionné une adresse IP le client envoi un datagramme ARP pour savoir si cette adresse est déjà prise par quelqu'un.
Cependant l'utilisation de ARP peut engendrer des conflits dans le cas de deux réseaux interconnectés.
Pour cela les adresses de l'intervalle 169.254.0.1 à 169.254.255.254 sont non-routables (le routeur ne laisse pas passer les paquets broadcasts, utilisés par ARP) ce qui restreint l'utilisation de AutoIP à un cadre de petits réseaux.

Néanmoins une variante étendue de ce mécanisme est proposée comme une alternative au DHCP dont la mise en place dans de grands réseaux industriels peut entrainer d'importants coûts afin de disposer de routeurs compatibles pour le fonctionnement de l'Agent Relais. Cette extension de AutoIP, garantit des coûts de maintenance moindres et fournit une automatisation maximale de la configuration lors des changements physiques de dispositifs.

Chapitre 6

DHCP

DHCP signifie Dynamic Host Configuration Protocol. Ce protocole conçu en 1997 par K. Droms de l'Université de Bucknell, permet de configurer dynamiquement un ordirnateur sur un réseau TCP/IP, c'est à dire de lui attribuer une adresse IP libre, et éventuellement l'adresse d'une passerelle pour accéder à internet, les adresses des serveurs WINS ou DNS, etc.
Vous pouvez consulter l'annexe pour connaitre tous les paramètres qui peuvent être attribuer par un serveur DHCP.

DHCP à quoi ça sert ?

Le protocole DHCP sert à configurer automatiquement un ordinateur sur le réseau. Imaginez que le réseau ne possède pas de serveur DHCP et pas d'autres mécanismes d'autoconfiguration, l'administrateur réseau devrait configurer à la main chaque nouvel ordinateur souhaitant se connecter au réseau. Il doit alors tenir à jour la liste des ordinateurs avec leurs adresses IP pour éviter d'attribuer deux fois la même adresse IP. Imaginez maintenant que l'administrateur souhaite changer le serveur WINS d'un réseau de 200 machines : il devra passer sur les 200 machines pour modifier ce paramètre : travail laborieux !

6.1 Principe de fonctionnement

6.1.1 Le serveur DHCP

Le serveur DHCP doit permettre aux membres d'un réseau de pouvoir communiquer entre eux en assurant de distribuer des adresses diffèrentes pour chaque machine. Il doit fournir à ses clients, un minimu d'information : le masque du réseau et une adresse IP libre. Le serveur DHCP peut fournir de nombreuses informations (voir annexe pour la totalité des informations), comme par exemple l'adresse IP de la passerelle Internet. Ainsi les clients auront connaissance de l'IP de la passerelle et pourront accéder au NET.
Lorsqu'un client demande la configuration du réseau à un serveur DHCP, celui-ci lui transmet par un paquet unicast (pour l'instant le client ne possède pas d'adresse IP mais il a transmit son adresse MAC lors de sa requête).

6.1.2 Le client DHCP

Quand une machine est démarrée, elle n'a aucune information sur sa configuration réseau, et surtout, l'utilisateur ne doit rien faire de particulier pour trouver une adresse IP.
Pour que la machine soit configurée automatiquement, elle va envoyer un message à tout le monde présent sur le réseau et attendre la réponse d'un serveur DHCP. La technique utilisée est le broadcast : pour trouver et dialoguer avec un serveur DHCP, la machine va simplement émettre un paquet spécial de broadcast (broadcast sur 255.255.255.255 avec d'autres informations comme le type de requête, les ports de connexion...) sur le réseau local. Lorsque le serveur DHCP recevra le paquet de broadcast, il renverra un paquet unicast car cette fois le serveur connait l'adresse MAC du client, contenant toutes les informations requises pour le client.

La coexistance de plusieurs serveur DHCP est assez périlleux pour le bon fonctionnement du réseau, mais si le cas se présente et qu'un client souhaite obtenir les paramètres du réseau, la plupard du temps il ne communiquera qu'avec le premier serveur DHCP qui a répondu.

6.1.3 L'agent relais

Un routeur ne transmettant pas les broadcasts, il faut qu'ils soient transmis par quelqu'un si besoin est. C'est le rôle l'agent de relais.

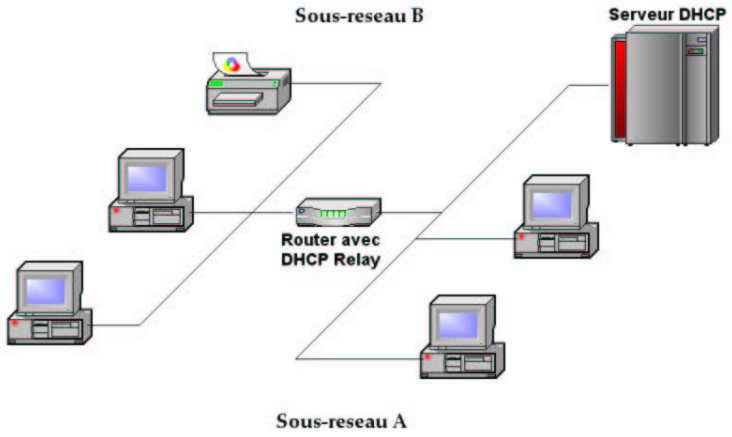


FIG. 6.1 – Agent relais

6.2 Exemple de connexion

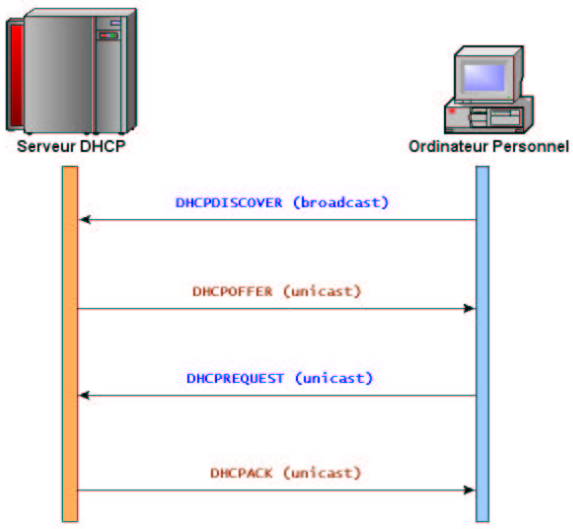


FIG. 6.2 – paquets échangés par le client et le serveur

6.3 Les paquets

Voici les différents paquets que peuvent s'échanger le client et le serveur.

Message	Utilisation
DHCPDISCOVER	Diffusion du client pour localiser les serveurs disponibles
DHCPOFFER	Du serveur au client pour répondre au DHCPDISCOVER avec les paramètres de configuration
DHCPREQUEST	Message client aux serveur soit : **(a)** qui demande les paramètres à un serveur et décline implicitement les offres de tous les autres, **(b)** qui confirme la validité des adresses précédement allouées, par ex : un redémarrage système, ou **(c)** qui étend le bail pour une adresse réseau en particulier
DHCPACK	Du serveur au client avec les paramètres de configuration et qui inclut l'adresse réseau déjà attribuée.
DHCPNAK	Du serveur au client indiquant que la notion d'un client pour les adresses réseau est incorrecte. (par ex. : si un client est déplacé sur un nouveau sous réseau) ou que le bail du client a expiré.
DHCPDECLINE	Client vers serveur indiquant que l'adresse réseau est déjà utilisée.
DHCPRELEASE	Client vers serveur libérant l'adresse réseau et annulant le bail.
DHCPINFORM	Client vers serveur, demandant seulement les paramètres de con figuration locaux ; le client possède déjà une adresse réseau attribué de manière externe.

TAB. 6.1 – tiré de la RFC-2131

6.4 Les avantages de ce protocole

Cet adressage dynamique réduit les problèmes de maintenance sur un réseau d'importance moyenne : toute nouvelle machine peut s'y intégrer rapidement. En outre, il est possible de communiquer d'autres informations de configuration dynamique du réseau via le mode DHCP, comme l'adresse du serveur DNS externe, par exemple. La configuration réseau des clients est ainsi mise à jour à chaque nouvelle connexion au serveur. Le paramétrage des propriétés réseau des clients est ainsi simplifié.
L'avantage majeur du DHCP est la configuration centralisée entraînant moins d'erreurs.

6.5 Les inconvénients de ce protocole

6.5.1 Problèmes divers

DHCP n'est pas compatible avec tous les systèmes d'exploitation. Son implantation difficile à travers les pare-feu.

6.5.2 Problèmes liés à la sécurité

Il est possible qu'un utilisateur malveillant ou inexpérimenté créer des perturbations sur le réseau en installant un serveur DHCP non-officiel.

Le problème est que le serveur pourrait attribuer des adresses Ip appartenants déjà à d'autres ordinateurs. Dans ce cas il est possible d'obtenir deux dispositifs ou plus avec la même addresse

IP.
Ces problèmes peuvent survenir à tout moment et le réseau sera innondé de collisions d'adresses IP ce qui implique un résultat souvent catastrophique.

D'autres problèmes sont possibles si le faux serveur DHCP fourni des configurations incorrectes ainsi qu'une periode de validité de l'adresse IP trop longue ou trop courte.
Par exemple, si les options 1 ou 3 [RFC2132] sont incorrectes (masque sous-réseau par exemple), le dispositif ne pourrait pas communiquer avec les ordinateurs extérieurs à son propre sous-reseau (ou dans certains cas, toute communication serait impossible).

Force est de considerer un scenario encore plus grave tel qu'un ordinateur qui charge son systeme d'exploitation à partir du reseau en utilisant le protocole TFTP. Si ce dispositif est dirigé pour charger un fichier, probablement sur un serveur différent il permet au malfaiteur de succéder le client, et donc d'avoir accés aux paramètres de démarrage contennant énormement d'informations vitales (mot de passes, fichiers d'initialisation etc ...).

A noter que l'utilisation du protocole BOOTP implique les mêmes vulnerabilités.

Deuxième partie

Proposition d'un protocole alternatif

Nous avons vu que DHCP n'avait pas vraiment de dispositifs en place pour faire face à une quelconque attaque. En effet, à l'époque de sa conception, les aspects sécurité d'un protocole ne faisaient pas toujours partie des principales préoccupations tout simplement parce que les initiateurs de certains protocoles n'imaginaient pas toujours que leur protocole pourraient avoir un succès en dehors de leur bureau ou université. Aujourd'hui, il en est tout autrement et cette sécurité est désormais un point très critique dans l'élaboration d'un nouveau protocole.

Dans cette partie, nous allons donc proposer un protocole sécurisé : le SDHCP[1]. Après l'avoir décrit en détails et expliqué sur quels mécanismes il s'appuie, nous verrons certaines attaques, et ce qui a été fait pour s'en protéger. Nous terminerons par l'étude de l'implémentation d'un petit prototype.

[1]SDHCP : Secure DHCP

Chapitre 7

Le protocole SDHCP

SDHCP se propose de fiabiliser la configuration d'un client grâce à une authentification de ce dernier avec login et mot de passe.

Seuls pourront donc se connecter, des clients possédant déjà un compte sur le serveur SDHCP. S'ils n'en possèdent pas, il devront en demander la création à l'administrateur du réseau.

Chacune des deux parties devra se plier à certaines exigeances :

- Le serveur devra être sûr de l'identité du client avant de lui envoyer ses paramètres de configuration.

- Le client, quant à lui, devra avoir confiance en les paramètres qu'il reçoit. Il doit être certain qu'ils n'ont été ni lus ni modifiés par un tiers. En effet, nous avons vu avec DHCP qu'une configuration issue d'un serveur malicieux pouvait avoir de graves conséquences pour la sécurité de la machine.

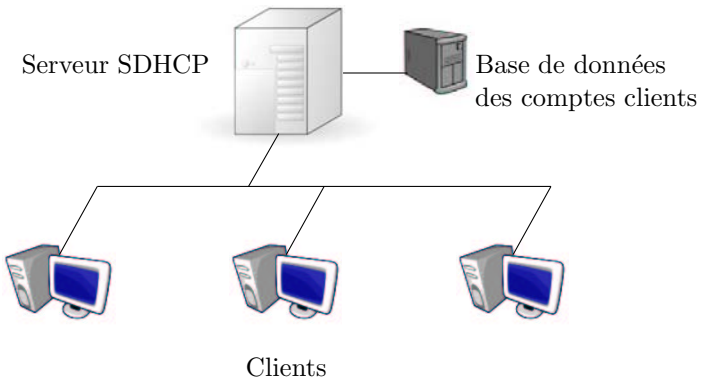


Fig. 7.1 – Exemple d'architecture avec un serveur SDHCP

7.1 Les choix techniques

Concernant la sécurité des échanges, nous nous baserons principalement sur deux systèmes.

Le chiffrement des informations s'effectuera avec l'algorithme RSA[1] tandis que le calcule d'empreintes se fera avec la fonction de hachage SHA[2].

7.1.1 RSA

RSA est un cryptosystème basé sur la difficulté que nous avons pour factoriser de grands nombres. Il utilise un algorithme dit "à clé publique" car il utilise deux clés. La première, publique, permet seulement de crypter un message. La seconde, privée, permet de le décrypter.

Un utilisateur qui souhaite qu'on lui envoie des messages cryptés pourra donc diffuser sa clé publique. Ses interlocuteurs lui enverront des messages chiffrés que seul notre utilisateur pourra déchiffrer car il possède la clé privée associée.

Avec RSA, client et serveur pourront ainsi s'échanger leurs clés publiques afin de s'envoyer des messages cryptés qui ne pourront être lus par d'autres personnes.

Pour l'implémentation, nous préconisons l'utilisation d'une clé d'au moins 1024 bits. En effet, casser une clé de 256 bits peu se faire en moins d'une heure avec un ordinateur personnel, et il a été annoncé il y peu qu'une clé de 512 bits avait été cassée avec l'aide d'un gros calculateur.

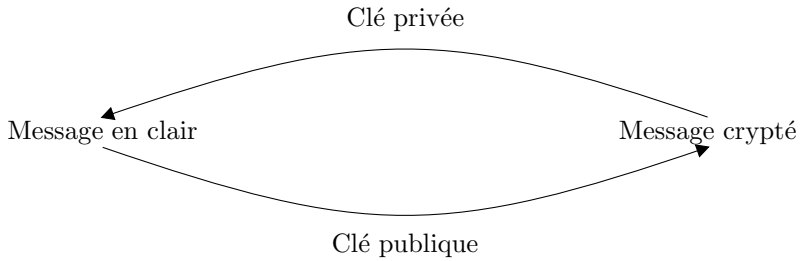


FIG. 7.2 – Fonctionnement du RSA

7.1.2 SHA

SHA est une fonction de hachage qui permet de calculer l'empreinte d'un message donné, c'est une évolution du mécanisme similaire MD5[3]. Une empreinte a une taille fixe, elle ne dépend pas de la longueur du message. La sécurité de ce sytème est basée sur le fait que l'on ne peut retrouver le message d'origine à partir de son empreinte. SHA est une évolution plus sûre du mécanisme similaire MD5[4]. Ce système permettra aux deux parties de montrer qu'elles ont connaissance d'une même information sans pour autant se la transmettre via le réseau, qui est par définition, un média peu sûr.

Prenons l'exemple d'un client qui possède un compte sur un serveur. Le serveur connait le mot de passe de son client, et ce client aimerait bien prouver au serveur qu'il connait ce mot de passe mais, pour des raisons de sécurité, il préfère éviter de le faire transiter sur le réseau. Le client envoie alors l'empreinte de son mot de passe au serveur qui va la comparer avec celle qu'il possède (en pratique, pour des raisons de sécurité, le serveur ne conserve en mémoire que cette empreinte). Il pourra ainsi déterminer si oui ou non le client possède le bon mot de passe. Si la

[1] RSA : Rivest, Shamir et Adleman, les noms des auteurs
[2] SHA : Secure Hash Algorithm
[3] MD5 : Message Digest 5
[4] MD5 : Message Digest 5

valeur de cette empreinte a été interceptée par un tiers, il lui est impossible de déterminer le mot de passe d'origine grâce aux propriétés de cette fonction de hachage.

On notera que ce type d'utilisation est souvent couplée à un marquage de l'empreinte pour éviter toute attaque par rejeu, comme nous le verrons au chapitre 8.

La déclinaison SHA1 donne par exemple des empreintes de 160 bits.

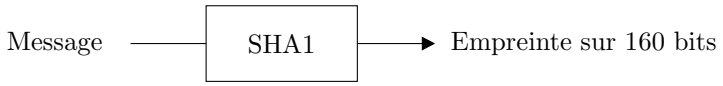


FIG. 7.3 – Fonctionnement du SHA

7.2 Format d'un paquet SDHCP

Tout comme pour DHCP, un paquet SDHCP sera encapsulé dans un datagramme UDP.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     op (1)    |  taille (1)   |            xid (2)            |
+---------------+---------------+-------------------------------+
|                                                               |
|                        option (variable)                      |
+---------------------------------------------------------------+
```

Champs	Octets	Descriptions
op	1	Type de message
taille	1	Taille du paquet SDHCP
xid	2	Numéro de transaction, définit au premier paquet de façon aléatoire, par le client
options	var	Suite d'options dépendantes du type de message

Format d'une option

```
 Code     Long.   Information
+-----+----------+---
|  x  |     n    |   ...
+-----+----------+---

Code    Numero de l'option (codee sur 1 octet)
Long.   Taille de l'information qui suit, sur 2 octets
Info.   Information dont la longueur est definie par Long.
```

Code	Description
1	Bits pairs/impairs d'un paquet
2	Nombre aléatoire de sécurité
4	Login
5	Empreinte SHA
6	Paramètres de configuration
10	Clé RSA n
11	Clé RSA e
255	Marqueur de fin d'options

7.3 Diagramme de séquence

La configuration d'un client avec SDHCP s'effectue en 8 étapes, là ou DHCP n'en prenait que 4. Ceci est dût aux différentes protections mises en place pour sécuriser le protocole.

- Client et serveur commencent par s'échanger leurs clés publiques, ce qui leur permettra ensuite de s'envoyer des messages chiffrés.
- Ils s'échangent ensuite deux numéros de sécurité qui permettront d'éviter une attaque par rejeu, comme nous le verrons au chapitre 8.
- Pour continuer, le client envoie ses login et mot de passe au serveur.
- Si tout va bien, le serveur retourne alors la configuration du client.

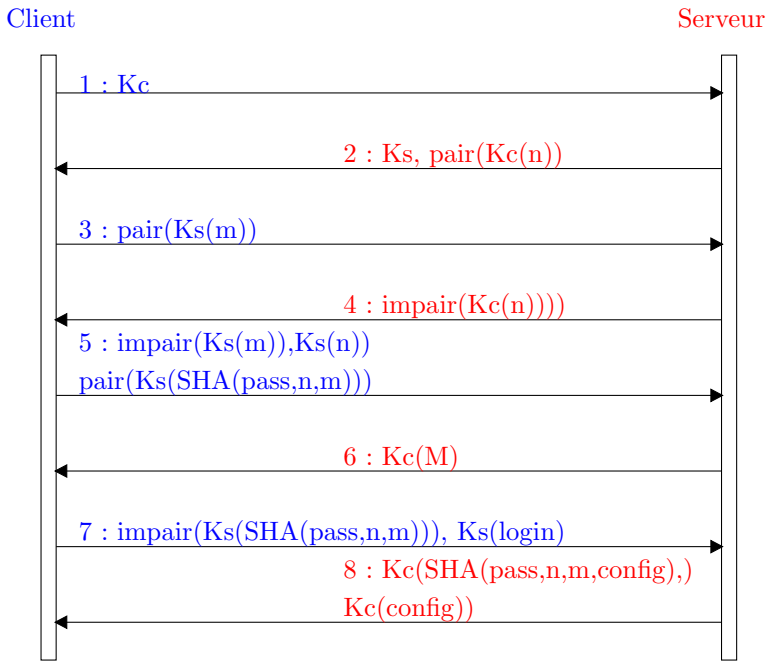


FIG. 7.4 – Diagramme de séquence des messages échangés entre le client et le serveur SDHCP

Vous pouvez remarquer que l'ensemble des ces echanges sont imbriqués. Seule une moitié du message est envoyée à la fois. Ceci permet d'éviter l'attaque dite "du passeur de seau", que nous verrons aussi au chapitre 8.

7.4 Les messages SDHCP

1. Kc
   Le client diffuse sa clé publique sur le réseau en effectuant un broadcast.

2. Ks, pair(Kc(n))
   Le serveur retourne sa clé publique et la moitié d'un message crypté avec la clé du client. Ce message représente un nombre aléatoire n de 64 bits.

3. pair(Ks(m))
   Le client envoie lui aussi la moitié d'un nombre aléatoire m aléatoire crypté avec la clé du serveur.

4. impair(Kc(n))
   Le serveur envoie le reste de ce nombre n.

5. impair(Ks(m)), Ks(n), pair(Ks(SHA(pass,n,m)))
   Le client envoie le reste de son nombre m, le nombre n du serveur pour lui montrer qu'il l'a bien lu, et la première partie de son mot de passe concaténés avec les deux nombres de sécurité.
6. Ks(m)
   Le serveur prouve au client qu'il a bien reçu et lu le nombre m en le lui renvoyant.

7. impair(Ks(SHA(pass,n,m))), Ks(login)
   Le client envoie le reste de ses paramètres d'authentification, dont le login.

8. Kc(SHA(pass,n,m,config), paramètres du réseau)
   Après avoir authentifié le client, le serveur lui retourne ses paramètres de configuration. Il lui donne aussi l'empreinte de son mot de passe concaténé avec les numéros de sécurité et les paramètres réseau, ce qui prouve à notre client qu'il a bien à faire avec le vrai serveur car seul lui peut construire cette empreinte étant donné qu'il est seul à avoir accès à la base de données des comptes clients. En faisant intervenir les paramètres réseau dans le calcul de l'empreinte, on garantie du même coup l'intégrité des informations contenues dans cette configuration.

Chapitre 8

Les attaques

Dans ce chapitre, nous allons présenter quelques unes des attaques possibles sur notre protocole. Nous verrons leurs principes et implications, ainsi que les précautions qui ont été prises pour nous en protéger.

8.1 Le rejeu

Le principe

Imaginons qu'un client souhaite se connecter à un serveur où il possède un compte, accessible par un login et un mot de passe.

Notre client, envoie donc son login à notre serveur qui lui réclame alors l'empreinte du mot de passe associé. En effet, si quelqu'un écoute, on préfère éviter de transférer le mot de passe. L'empreinte suffit à s'authentifier.
Le serveur accepte alors la connexion si l'empreinte reçue correspond bien à celle dont il dispose, le client fait ses petites affaires, puis finit par se déconnecter.

Considérons maintenant qu'il y avait un intrus qui écoutait toute la conversation. Il possède donc le login, et l'empreinte du mot de passe. Avec ceci, il peut se connecter au serveur en se faisant passer pour le client.
Il envoie ainsi au serveur le login de ce client, suivit de l'empreinte du mot de passe qu'il avait intercepté. Le serveur constate alors que l'empreinte correspond bien au login et donne donc accès à notre intrus.

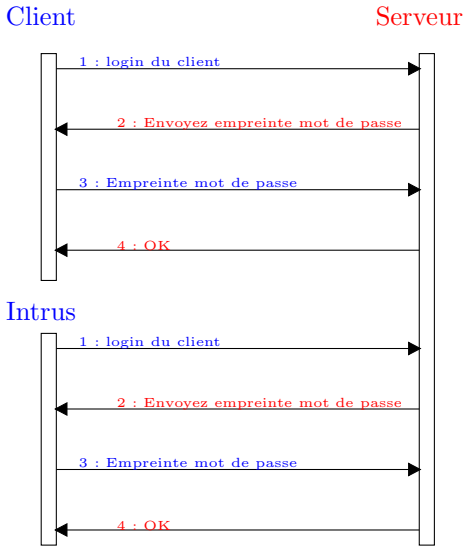


FIG. 8.1 – Principe d'une attaque par rejeu

Comment s'en protéger ?

Il existe plusieurs façons de contrer ce type d'attaque. La plus naïve consiste à dater les messages envoyés et à ne pas tenir compte d'un message trop vieux. Le problème, c'est que les horloges d'un réseau ne sont jamais bien synchronisées. et qu'on devra toujours laisser un intervalle de validité au message, intervalle pendant lequel un intrus peut le rejouer.

La technique que nous avons utilisé consiste à marquer les messages avec deux nombres aléatoires de 64 bits. L'un choisi par le client, l'autre par le serveur. Ces deux nombres seront définits pour tout la durée de la session. L'empreinte tiendra alors compte de ces deux nombres. Du coup, pour un même mot de passe, nous aurons une empreinte différente à chaque session. Le rejeu n'est alors plus possible.

8.2 Le passeur de seau

Le principe

Imaginons que l'on souhaite faire de l'authentification en utilisant la cryptographie à clé publique. A et B souhaiteraient discuter entre eux, mais s'assurer d'abord mutuellement de leur identité.

Supposons dans un premier temps qu'ils connaissent chacun la clé publique de l'autre.
A commence par choisir un nombre aléatoire n, et l'envoie à B, chiffré avec sa clé publique.
B reçoit le message, et le décrypte avec sa clé privée. Il choisit alors lui aussi un nombre aléatoire m qu'il envoie crypté à A, en prenant soin de joindre n.
A constate alors que B a pu lire son nombre n et il est donc convaincu de l'identité de ce dernier. Pour prouver lui aussi son identité à B, il lui retourne son nombre m.

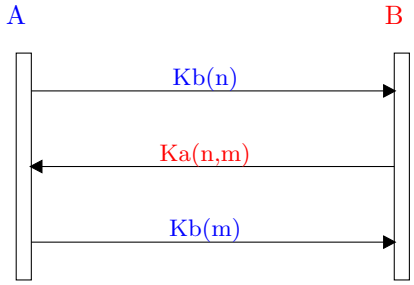


FIG. 8.2 – Principe de l'authentification à l'aide de la cryptographia à clé publique

Comment notre intrus peut-il détourner ce protocole à son avantage ?

Notre protocole a un inconvénient majeur : il suppose que A et B connaissent chacun la clé publique de l'autre. Si ce n'est pas le cas, ils vont devoir se l'échanger au préalable et c'est là que notre intrus passeur de seau intervient.

L'intrus se placera au milieu en se faisant passer pour B auprès de A et pour A auprès de B. Pour cela, au moment de l'échange des clés entre A et B, notre intrus I interceptera les messages et donnera sa propre clé publique.

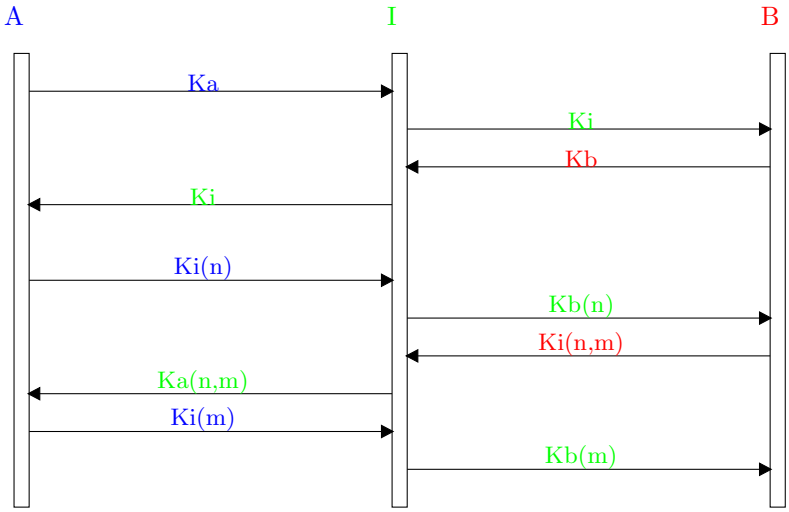


FIG. 8.3 – Principe de l'attaque du passeur de seau

Ce type d'attaque est très grave car ici, l'intrus peut lire absolument tous les messages echangés, pire, il pourra même les modifier.

Comment s'en protéger ?

En 1984, Rivest et Shamir (les R et S de RSA), ont proposé un protocole qui déjoue ce type d'attaque : *le protocole imbriqué.*

Après l'échange des clés publiques, A n'envoie qu'une moitié de message à B, par exemple les bits pairs (après chiffrement), B répondant alors avec ses bits pairs. Après avoir reçu les bits pairs de B, A envoie ses bits impairs, ce que fait également B tout de suite après.

L'astuce ici est que lorsque I prend les bits pairs de A, il ne peut déchiffrer le message, même si l'on imagine qu'il a la clé privée. Il ne peut donc rechiffrer les bits pairs en utilisant la clé publique de B. S'il envoie n'importe quoi à B, le protocole va continuer et B verra rapidement que le message complet n'a aucun sens. Il réalisera alors qu'il a été trompé.

C'est cette technique que nous utilisons dans SDHCP.

8.3 Le faux serveur SDHCP

Le principe

Considérons maintenant que l'attaquant souhaite tromper les clients en se faisant passer pour un serveur SDHCP. Il pourrait par exemple leur donner des paramètres de configuration érronés dans le but de les rendre inopérants sur le réseau, ou tenter de détourner une liaison afin de l'espionner.

Le faux serveur effectuerait toute la transaction avec le client, et à la fin, lui donnerait des paramètres de connexion quels que soient les login et mot de passe donnés par ce dernier.

Comment s'en protéger

Pour éviter une telle attaque, il faut réfléchir aux différences entre un serveur officiel et un faux serveur : l'accès à la base de données des comptes clients. Ici, en utilisant cette base de données, nous retournons une information que seul le vrai serveur peut connaitre. C'est le rôle du dernier paquet retourné par le serveur : *Kc(SHA(pass,n,m,config), paramètres du réseau)*
En effet, celui çi contient l'empreinte du mot de passe du client avec les numéros de sécurité. Ainsi, le client ne trouvant pas une empreinte correspondant à son mot de passe comprendra que la configuration reçue ne provient pas du serveur officiel du réseau.

Chapitre 9

Implémentation d'un prototype

Afin de tester ce protocole, nous avons mis au point un prototype simpliste, mais permettant de nous faire toucher du doigts certaines ambiguïtés et insuffisances qui existaient dans la première version de SDHCP.
Ainsi, la version actuelle du serveur n'accepte qu'un client à la fois, et communique avec une base de données fictive dans laquelle il n'y a un qu'un seul compte client.
Néanmoins, une version "complète" et utilisable sera développée prochainement afin de tester ce protocole en grandeur nature.

9.1 Fonctionnement de l'application

L'application réalisée est constituée de deux exécutables, *sdhcpd* pour le serveur et *sdhcp* pour le client auquel il faut passer en paramètres un *login* et un *mot de passe*.

Au lancement, client et serveur créent chacun une nouvelle clé RSA de 1024 bits (taille minimale préconisée pour un peu de sécurité). Le serveur se met ensuite directement en attente d'une connexion et le client commence immédiatement la session en diffusant sa clé publique sur le réseau.

Exemple de trace d'exécution

Coté client :

```
zkma@gandalf:~/Projets/TE/Src$ ./sdhcp
Usage : ./sdhcp login password

// Essai avec un mot de passe correcte...
zkma@gandalf:~/Projets/TE/Src$ ./sdhcp zkma toto
Recherche du serveur SDHCP...
Authentification...
Configuration : IP:192.168.1.1 MASQUE:255.255.255.0
Client configur\'e.

// ... et avec un mot de passe errone
zkma@gandalf:~/Projets/TE/Src$ ./sdhcp zkma toti
Recherche du serveur SDHCP...
Authentification...

zkma@gandalf:~/Projets/TE/Src$
```

Coté serveur :

```
zkma@gandalf:~/Projets/TE/Src$ ./sdhcpd
Lancement du serveur...

Creation d'une cle RSA de 1024 bits...

Attente d'un client...
Etape 1 : Reception de Kc
Etape 2 : Envoie de Ks, pair(Kc(n))
Etape 3 : Reception de pair(Ks(m))
Etape 4 : Envoie de impair(Kc(n))
Etape 5 : Reception de impair(Ks(m)), Ks(n), pair(Ks(SHA(pass,n,m)))
Etape 6 : Envoie de Kc(m)
Etape 7 : Reception de Ks(login), impair(Ks(SHA(pass,n,m)))
Client authentifie, envoie de la configuration
Etape 8 : Envoie de Kc(SHA(pass,n,m,config)), Kc(config)

Attente d'un client...
Etape 1 : Reception de Kc
Etape 2 : Envoie de Ks, pair(Kc(n))
Etape 3 : Reception de pair(Ks(m))
Etape 4 : Envoie de impair(Kc(n))
```

```
Etape 5 : Reception de impair(Ks(m)), Ks(n), pair(Ks(SHA(pass,n,m)))
Etape 6 : Envoie de Kc(m)
Etape 7 : Reception de Ks(login), impair(Ks(SHA(pass,n,m)))
Authentification du client incorrecte.

Attente d'un client...
```

9.2 Etude des choix techniques

Le choix du langage de développement pour cette implémentation s'est porté sur le *C* qui dispose d'une bonne couche réseau, et de toutes les librairies nécessaires pour la cryptographie, notamment avec *OpenSSL*.

9.2.1 La librairie OpenSSL

OpenSSL[1] propose une grande quantité de fonctions cryptographiques utilisées pour la sécurité des réseaux.
Nous l'avons utilisée pour le cryptage *RSA* et pour sa fonction de hachage *SHA*, mais il existe aussi des fonctions pour utiliser les algorithmes *DES*, *3DES*, *IDEA* ou encore *MD5*. Vous l'aurez compris, c'est une librairie très complète.

RSA

L'utilisation de *RSA* dans un programme est guidée par un grand nombre de fonctions dont certaines avec des noms très obscures. Heureusement, il est possible de se restreindre à n'utiliser que quelques fonctions simples :

- *RSA_new()* permet de créer une structure *RSA*, vide dans un premier temps, qui permettra de stocker un couple clé publique/clé privée.
- *RSA_generate_key()* va remplir la structure précédemment créée en générant de façon aléatoire une nouvelle clé RSA.
- *RSA_public_encrypt()* chiffrera un message en utilisant la clé publique.
- *RSA_private_encrypt()* déchiffrera un message crypté avec la fonction précédente.

On peut noter que la structure *RSA* peut être remplie de façon manuelle, comme nous le faisons lorsque l'on reçoit une clé publique en ne renseignant que certains champs, ou de façon automatique via la fonction *RSA_generate_key()*.

SHA

Le calcule d'une empreinte SHA peut se faire via une fonction très simple : *SHA1()*. Celle-çi se base sur l'algorithme SHA-1 qui donne une empreinte de 160 bits.

Un groupe qui a pour vocation de tenter de casser des codes SHA a annoncé il y a quelques mois qu'une collision (ie. trouver deux chaines donnant une même empreinte) pouvait être trouvée en 2^69 opérations, ce qui reste encore hors de portée des moyens techniques actuels. Néanmoins l'utilisation des évolutions telles que SHA256 ou même SHA512, qui génèrent respectivement des empreintes de 256 et 512 bits, permettrait d'acrître la sécurité des empreintes sur le long terme.

Afin de tester les performances de cette fonction avec de petites chaînes, nous avons écrit un petit programme permettant de mesurer le temps mis pour effectuer 1 000 000 d'empreintes de messages d'une longueur moyenne de 40 caractères.
Les résultats sont excellents : moins de deux secondes sur un Intel Pentium 3 à 733 Mhz.

[1] OpenSSL : Open Secure Sockets Layer

9.3 Conclusion sur ce prototype

Ce prototype nous a permis de remarquer certains problèmes au sein de la première definition du protocole.

Ainsi, les formats des options, autrefois calqué sur celui utilisé par DHCP a été revu en permettant de gérer désormais des options d'une taille de 65535 octets, contre 256 auparavant. Cette dernière taille était en effet beaucoup trop limitée sachant qu'une clé RSA peut facilement dépasser cette borne.

Un champs *taille* a aussi été ajouté afin de faciliter la gestion des paquets, en particulier pour le serveur. Cette information permettra de découper facilemnet les paquets lus sur la socket dans le cas où plusieurs messages de différents clients sont arrivés en même temps.

Une expérimentation à plus grande echelle permettra sans doute de trouver de nouvelles améliorations à ce jeune protocole.

Bibliographie

[1] Andrew Tanenbaum - *Réseaux* - Dunod, 1999

[2] Gilles Dubertret - *Initiation à la cryptographie* - Vuibert, 1998

[3] Joe Casad, Bob Willsey - *TCP/IP* - CampusPress, 1999

[4] *http ://www.rfc-editor.org*

[5] *http ://fr.wikipedia.org/wiki/Accueil*

[6] Denis Chalon, Yves Durand, Bruno Richard - *An Overview of Automatic Network Configuration for IPv4 Appliances* - HP Labs Grenoble, 2004

[7] Oscar Cepeda, Bob Chambers - *Beyond DHCP - Work Your TCP.IP Internetwork With Dynamic IP*

[8] Andy Swales - *IP Address Assignment in Large Industrial Networks* - Network Vision, Inc.

[9] *Understanding DHCP and DNS* - Cisco, 2003

[10] Jacques Demerjian - *Thèse : Services d'autorisation et intégration au protocole d'attribution dynamique des adresses* - ENST, 2004

[11] *Must-Have Reference for Network* - Anritsu Corp., 2004