Diagnostic de sécurité des serveurs VoIP

Ravonimanantsoa N.M.V[1], Randriamitantsoa A. A[2] , Rakotondraina T.E[3]

Equipe d'Accueil Télécommunication, Images, Automatique (EA-TIA)

École Doctoral Télécommunication et Électronique

[1]ndaohialy@gmail.com, [2]andriau@hotmail.com,[3]tahiana.ezechiel@gmail.com

Résumé

L'utilisation de la VoIP dans le monde de la téléphonie devient de plus en plus fréquente. Mais l'architecture internet représente beaucoup de vulnérabilité du point de vue sécurité. Cet article nous montre l'utilisation abusive de la mémoire par les appels Sip ainsi que le risque éventuel de ces appels, et les mesures à prendre pour limiter ces risques

Mot clé

Sip ; DoS; astérisque; VoIP

Abstract

The use of VoIP in the telephony world is becoming more common. But the Internet architecture is a lot of vulnerability in terms of security. This article shows the memory usage for SIP calls and the potential risk of these calls, and the measures to be taken to mitigate these risks

Keyword

Sip; DoS; astérisque; VoIP

1. Introduction

La Voix sur IP est une technologie qui représente un énorme avenir sur le mode de communication entre humains. La téléphonie analogique traditionnelle à base de technologies est en train de perdre une part de marché que la téléphonie basée sur des réseaux de paquets. Ce dernier devrait poursuivre sa croissance tant dans les réseaux d'entreprise tant dans ceux du public. Par conséquent, les paquets voix produits lors de conversations téléphoniques sont nettement supérieurs à celle des données qui transite les réseaux IP. La qualité adéquate de l'assurance de services pour le transport de réseau IP la voix en paquets IP (VoIP) nécessite la connaissance de données caractéristiques, qui sont envoyées à travers ces réseaux. Comme toute conversation téléphonie traditionnelle, la voix sur IP peut être considérée comme un processus normal qui associe la conversation entre deux ou plusieurs interlocuteurs.

Par ailleurs, l'interlocuteur parle fait souvent de petites lacunes entre le langage parlé des phrases, des mots et des syllabes. En outre, voix dans un codeur est également compressé permettent de réduire la consommation de bande passante dans réseau de paquets le transport de paquets VoIP. Afin de garantir la qualité de service pour un utilisateur et simultanément de profiter pleinement de réseau disponible bande passante, il est nécessaire de déterminer son utilisation distribution au cours de la conversation. Les recherches, qui tentent de déterminer cette distribution, besoin d'un modèle source approprier VoIP, dont le principal élément c'est le protocole.

Afin de faciliter la construction d'une QoS (quality of service) bien définie, la présentation d'un modèle d'un protocole qui est exploité par la VoIP doit être étudiée au niveau du réseau internet. Or dans le réseau internet, beaucoup d'architecture représente beaucoup de vulnérabilité comme de DOS (Deny of

service):les serveurs web, les serveurs de messagerie, mais aussi les serveurs VoIP[1]. Ces services qui se reposent sur l'architecture client-serveur ont l'obligation de servir leur client et ces clients accèdent concurremment à un service en ligne fourni par un serveur [2][3][5]. Un serveur qui soumit à une plus forte charge risque de ne plus répondre à la différente requête suivante. Dans le cas extrême, une forte charge peut provoquer une indisponibilité.

Beaucoup de technique est proposé pour éviter l'écroulement d'un serveur. Pour notre cas, après les expériences nous avons pu constater que chaque appel Sip représente un risque pour le serveur et cela se situe au niveau matériel ou plus précisément au niveau du mémoire

Dans cet article, nous allons donc voir l'état de notre serveur pour chaque appel Sip et nous allons propose une solution pour remédier les risques.

2. Généralité sur le protocole SIP

2.1 Structure d'un appel SIP

SIP est un protocole d'application qui permet d'établir, modifier et terminer des sessions multimédias, en particulier des appels téléphoniques sur IP. Cette tâche est compliquée par le fait qui les utilisateurs peuvent utiliser différents terminaux pour communiquer, utiliser différents médias voire utiliser différents noms. SIP offre deux mécanismes principaux pour gérer la mobilité des utilisateurs :

- Localisation des utilisateurs : SIP permet de découvrir la localisation actuelle (terminal) d'un utilisateur, c'est-à-dire la traduction entre le nom d'un utilisateur et son adresse IP actuelle.
- Négociation des paramètres de session : après la découverte, les utilisateurs utilisent SIP pour négocier les paramètres de la session, en particulier le codage des flux média à établir.

2.2 Les composants d'un appel SIP

La meilleure façon d'étudier le fonctionnement du protocole SIP est de décrire les scénarios d'utilisation les plus fréquents.

Le scénario le plus simple de l'établissement d'un appel est montré à l'Illustration deux téléphones, A et B, établissent un appel sont utiliser un proxy SIP. Ce scénario est peu réaliste, puisque le téléphone A doit connaître l'adresse IP de B, ce qui n'est généralement pas le cas.

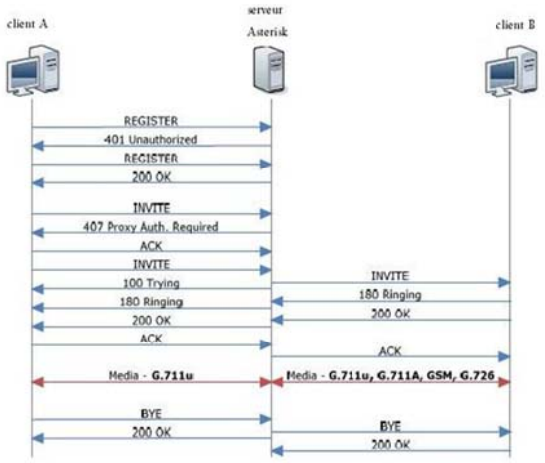

Fig1.*Composante d'un appel sip*

SIP fonctionne d'après un modèle « requête – réponse » similaire à HTTP. Une requête invoque une méthode et déclenche une action sur le serveur ou une réponse.

L'établissement direct d'un appel commence avec un message INVITE envoyé de l'appelant à l'appelé. Il s'agit donc d'une requête SIP qui contient la méthode INVITE et qui déclenche la fonction d'établissement d'appel. En plus des paramètres SIP, ce message INVITE contient la description du flux média qu'A aimerait établir.

Le téléphone B qui reçoit cette requête va sonner pour avertir l'utilisateur de l'appel entrant. Le téléphone répond à la requête avec une réponse provisoire RINGING, qui sert à informer A que l'établissement de l'appel est en cours.

Le téléphone A qui reçoit cette réponse peut en informer l'utilisateur avec la tonalité idoine ou l'affichage d'un message.

Dans notre cas, B accepte l'appel. Quand il décroche, son téléphone envoie une réponse finale

OK qui indique à A que l'appel a été accepté. Cette réponse contient également une description du flux média que B aimerait établir avec A.

Finalement, A confirme la réception de la réponse finale avec un message ACK. Ce message conclut l'établissement de l'appel. Chaque participant connaît maintenant l'adresse IP de l'autre et chacun a annoncé les caractéristiques du flux média à établir.

Un flux média bidirectionnel est établi entre les téléphones, typiquement en transportant les paquets voix sur RTP et UDP.

Pendant la session RTP, les deux participants peuvent modifier les paramètres du flux.

Ceci peut se faire simplement en envoyant une nouvelle requête INVITE avec les paramètres modifiés. L'autre participant répond comme avant avec une réponse OK et le demandeur confirme le changement avec un message ACK.

À la fin de l'appel, A raccroche ce qui déclenche une requête BYE. B confirme la terminaison de l'appel avec une réponse finale OK. À ce moment, le flux média, RTP est également terminé.

Contrairement à l'établissement avec INVITE, aucun message ACK n'est envoyé après la réponse OK. Ce traitement particulier des messages INVITE est dû entre autres au délai possible entre les messages INVITE et OK. Si B accepte l'appel après un délai long, A pourrait ne plus être en mesure d'établir l'appel.

2.3 Format des messages SIP

SIP est un protocole oriente texte, similaire à HTTP. Les messages sont codés en UTF-8 et peuvent donc contenir des caractères non-ASCII comme des caractères accentués.

Un message SIP contient en générale trois éléments :

- une ligne de départ qui indique le type du message,
- un en-tête du message, comprenant plusieurs lignes de texte,
- éventuellement un contenu du message, avec des informations supplémentaires.
- SIP connaît deux types de messages :
- les requêtes : envoyées d'un client à un serveur,
- les réponses : envoyés d'un serveur à un client.
- Un serveur peut être équipement de réseau comme un proxy, mais aussi un téléphone qui reçoit un appel (UAS).

```
INVITE sip:bob@192.168.1.2 SIP/2.0

Via: SIP/2.0/UDP 192.168.1.195;branch=z9hG4bKPjP1IqVNW6SRXf

Max-Forwards: 70

From: “Alice” <sip:alice@192.168.1.195>;tag=8-bHe3dXt.Xy1HJ2rYDKKHOo22-hYQo4

To: “Bob” <sip:bob@192.168.1.2>

Contact: <sip:alice@192.168.1.195>

Call-ID: mc5-M4FkXLuhYDiZAE@192.168.1.195

CSeq: 10414 INVITE

Content-Type: application/sdp

Content-Length: 462
```

Fig2. *Extrait d’une session SIP*

3. Environnement d’analyse

3.1 Méthodologie d’analyse

Le principal but de notre recherche c’est de faire une analyse du comportement du serveur VoIP[6] lors des appels Sip. L'évaluation de ces comportements va nous conduire à un résultat qui va être commenté dans le domaine de la sécurité.

Le second but c’est de proposer un moyen pour éviter le débordement de notre serveur.

3.2 Les outils utilisés pour les mesures

Les outils que nous avons utilisés sont tous des outils libres et aussi populaires.

Pour les simulations : nous avons utilisé comme serveur un PC Dell optiplex GX110 qui a comme ressource un processeur Pentium III de 863.872 MHz et une mémoire RAM de 256Mo, pour tout notre client nous avons aussi la même configuration sauf que la mémoire c'est de 128Mo. Le serveur est équipé d'un système d'exploitation Debian Lenny [7]et le serveur VoIP qu'on a choisi c'est Asterisk tandis que les postes clients sont tous équipés d'un système d'exploitation Windows XP Sp2 et on a installé le softphone X-lite pour les besoins de simulation des appels.

3.3 Architecture

Comme indique ci-dessus notre architecture se base sur une architecture client-serveur et notre client augment au fur et à mesure de notre besoin

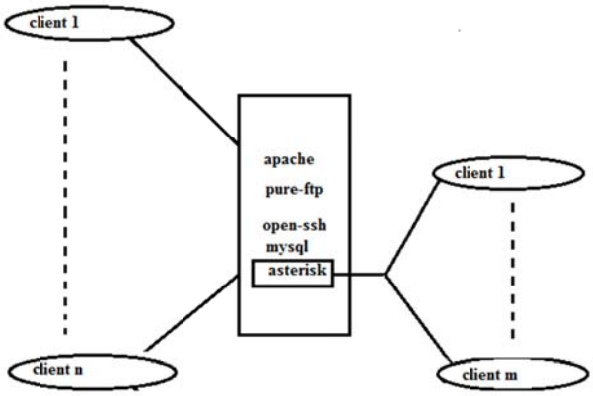

Fig3. *Architecture* **(01)**

3.4 Analyse

L'analyse proprement dite se déroule en trois grandes étapes et nous répétons les mêmes étapes au fur et à mesure que nous augmentons le nombre de machines (il s'agit du nombre de machines connectées physiquement au serveur).

L'utilisation des utilitaire comme « tcpdump » et « tcpstat » nous permet de voir ce qui se passe au niveau du réseau .ainsi les résultats de tcpstat sont capturés pour être tracé avec « gnuplot »

Étape 1 collecte des informations sur le système en lançant la commande « top » afin de bien définir sa stabilité vis-à-vis des charges qu'il va ensuite supporter (plusieurs requêtes vont être effectué durant toutes les expérimentations)

Étape 2 Lancements d'un appel sans que le correspondant ne décroche et collecte des informations sur le système en lançant la commande « top » afin de voir le comportement du serveur lors de cette initiation d'appel

Étape 3 le client B décroche et on collecte des informations sur le serveur pour déterminer le comportement la consommation en ressource de cet appel établi.

Ces trois étapes sont répétées au fur et à mesure que nous avons augmenté le nombre de clients.

4. Résultat et interprétation

4.1 Résultat du point de vue processeur et mémoire

Après plusieurs tests effectuer nous avons pu avoir les résultats suivant Figure,4,5,6

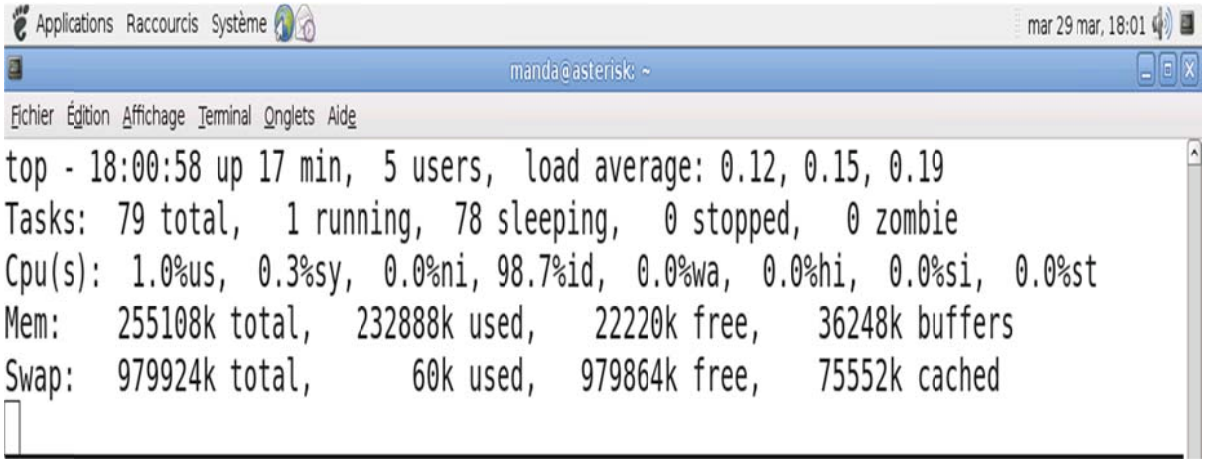

Fig4. *phase initialisation*

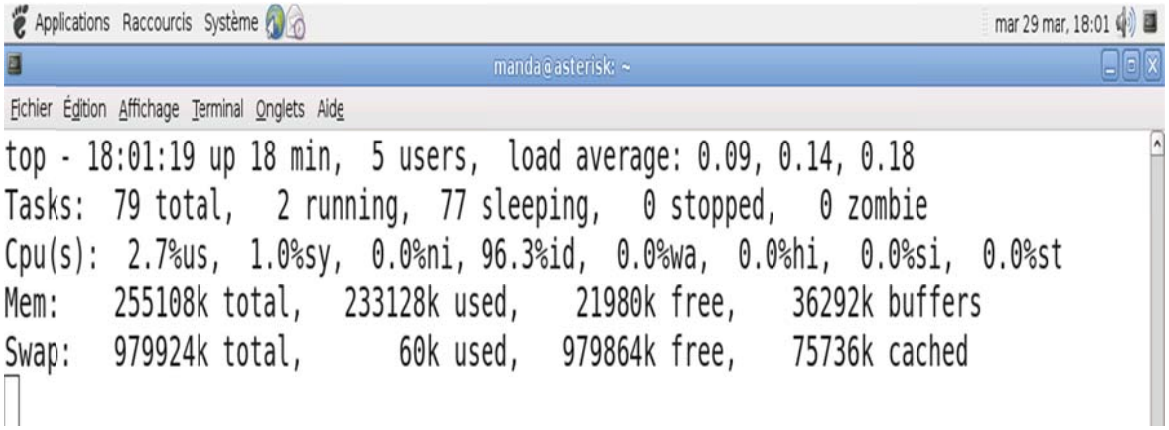

Fig5. *Émissions d'appel*

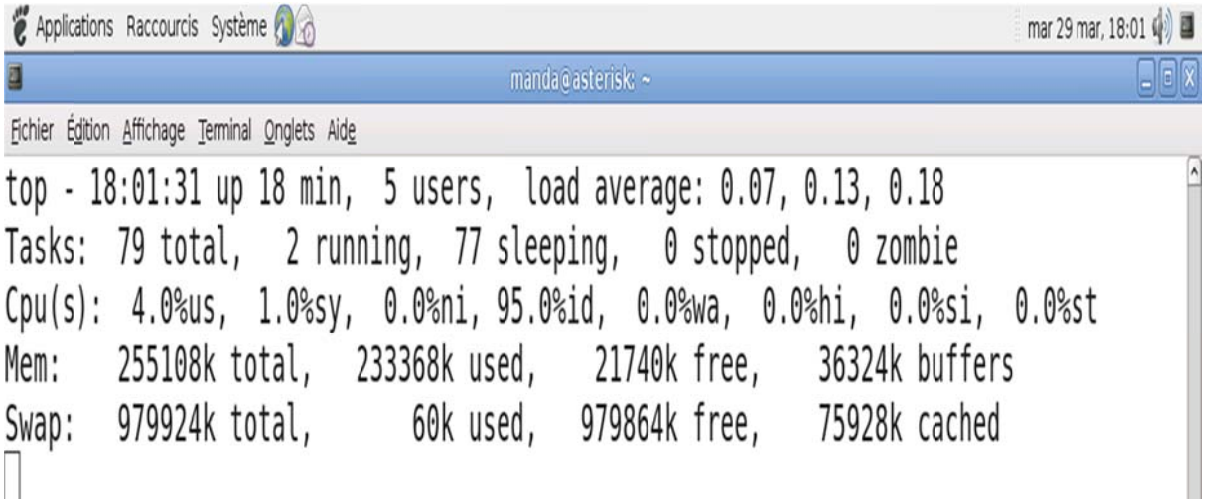

Fig6. *Phase établissement d'appel*

4.2 Interprétation au niveau du processeur et de la mémoire

D'après les simulations faites, lors de l'initiation de l'appel à partir d'un softphone, le niveau de la mémoire libre diminue. Ce qui explique que même si le client distant ne décroche pas le téléphone l'initiation de l'appel occupe déjà une partie de la mémoire. Et la suite de notre simulation ne fait que justifier tout ceci.

En augmentant le nombre d'appels, le niveau de la mémoire libre diminue au fur et à mesure (tableau 1) .nous avons donc comme déduction qu'un nombre suffisant d'appelants peut provoquer une indisponibilité du serveur parce que dans notre expérience nous constatons qu'un appel non reçu occupe 240ko de mémoire donc pour une mémoire de 256Mo il nous suffit d'un client nombre de clients de 1000 pour saturer la mémoire. Alors que la mémoire est utilisé par d'autre service des autres protocoles qui tourne sur le serveur comme le net bios ou encore le service qui lance le serveur proprement dit.

Si un client seul émet donc plusieurs appels sans réponse le principe du fil d'attente est donc casse parce que ces appels saturent la mémoire.

Dans notre expérience lorsque nous avons émis un nombre suffisant d'appels pour que la mémoire du serveur soit sature ,quand ce niveau est atteint , les autres clients qui essai de se communique trouve l'autre bout de la ligne occupe ce qui n'est pas vrai.

Client	Memory free (kb)	Client	Memory free (kb)
0	22220	6	20780
1	21980	7	20340
2	21740	8	20100
3	21500	9	19860
4	21260	10	19620
5	21020		

Tableau1. *Utilisation de la mémoire*

Résultat au niveau de l'interface réseau

Le scripte suivant nous a permet de tracer les trafics qui se produisent au niveau de l'interface réseau de notre serveur

```
#!/bin/bash

echo ce script permet de parser le flux tcpdump pour adapter à gnuplot

tcpstat -r traffic_2.dmp -o "%R\t%A\n" 60 > arp_2.data

tcpstat -r traffic_2.dmp -o "%R\t%C\n" 60 > icmp_2.data

tcpstat -r traffic_2.dmp -o "%R\t%T\n" 60 > tcp_2.data

tcpstat -r traffic_2.dmp -o "%R\t%U\n" 60 > udp_2.data

gnuplot gnuplot.script > graphe_2
```

Fig7. *Script pour le traçage avec gnuplot*

Le résultat au niveau de notre interface réseau nous est montré sur la figure de gnuplot ci-dessous ;

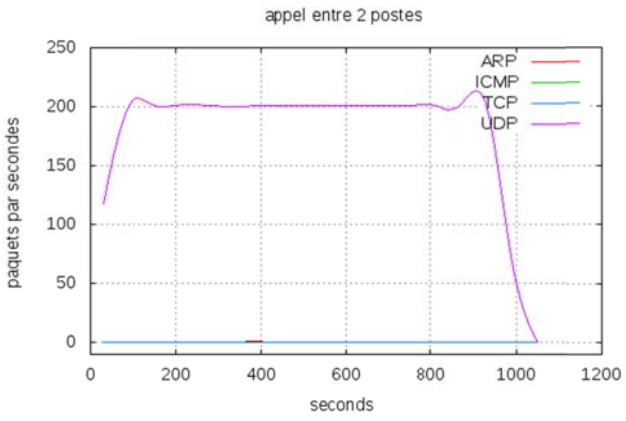

Fig8. *Graphe des paquets sur l'interface eth0*

Via: SIP/

14:53:15.036004 IP debian.local.sip > 192.168.1.113.21460: SIP, length: 500

E....C..@.-i...o...q..S....>SIP/2.0 100 Trying

Via: SIP/2.0/UDP 192.168.1.1

14:53:15.036535 IP debian.local.sip > 192.168.1.222.44094: SIP, length: 944

E...{...@.wU...o.....>...gINVITE sip:2222@192.168.1.222:44094;rinstance=0c

14:53:15.036616 IP debian.local.sip > 192.168.1.113.21460: SIP, length: 516

E.. .D..@.-X...o...q..S....NSIP/2.0 180 Ringing

Via: SIP/2.0/UDP 192.168.1.

Fig9. *Extrait de capture de tcpdump*

4.3 Interprétation au niveau de l'interface réseau

D'après nos constatations, nous avons remarqué que lors de l'initiation de l'appel ou SIP ouvre la session avec "Invité" on observe un pic dans le nombre de paquets par seconde, ce pic est montré à la clôture de la session peut être trouvée ou le message "BYE». Nous pouvons donc conclure que:

- Envoie le message invite un grand nombre de paquets de la conversation
- De même pour "bye"
- Enfin, notre simulation montre qu'un appel en absence peut provoquer une surcharge de notre interface réseau par ce pic

4.4 Proposition de solution

Dans notre cas tout d'abord nous avons mis quelque limite dans le fichier /etc/asterisk/asterisk.conf .c'est a dire le paramètre maxcalls et maxload

[options]

languageprefix = yes ; Use the new sound prefix path syntax

;verbose = 3

;debug = 3

;alwaysfork = yes ; same as -F at startup

;nofork = yes ; same as -f at startup

;quiet = yes ; same as -q at startup

;timestamp = yes ; same as -T at startup

;execincludes = yes ; support #exec in config files

;console = yes ; Run as console (same as -c at startup)

;highpriority = yes ; Run realtime priority (same as -p at startup)

;initcrypto = yes ; Initialize crypto keys (same as -i at startup)

;nocolor = yes ; Disable console colors

;dontwarn = yes ; Disable some warnings

;internal_timing = yes

systemname = my_system_name ; prefix uniqueid with a system name for global uniqueness issues

maxcalls = 10 ; Maximum amount of calls allowed

maxload = 0.9 ; Asterisk stops accepting new calls if the load average exceed this limit

Fig10.*:/etc/asterisk/asterisk.conf*

Ensuite nous avons limité l'accès au serveur proprement dit en éditant le fichier /etc/security/acces.conf[7]

```
User "" should be allowed to get access from hosts with ip addresses.

#Xxxxxxxxxxxxx

#xxxxxxxxx

#

# User "" should get access from network x.x.x.x

# This term will be evaluated by string matching.

# comment: It might be better to use network/netmask instead.

# User "" should be able to have access from domain.

# Uses string matching also.

+ : t : .x.x.x.x

# User "" should be denied to get access from all other sources.

- : : ALL

#

# User "essai" and members of netgroup "nis_group" should be

# allowed to get access from all sources.

# This will only work if netgroup service is available.

+ : @nis_group essai : ALL

#

# User "essai2" should get access from ipv4 net/mask

#+ : essai2: 127.0.0.0/24
```

Fig 6:*/etc/security/acces.conf*

Et enfin dans le fichier /etc/asterisk/user.conf nous avons changé le paramètre de callwaiting parce que ce paramètre permet à l'intention malveillante de créer une session inutile pour encombrer la ligne, mais aussi pour un mémoire insuffisant ce paramètre permet au client malveillant d'occuper le serveur tout en étant se comporter comme un simple client normal.

```
callwaiting = no

threewaycalling = no

callwaitingcallerid = yes

transfer = yes

canpark = yes

cancallforward = yes

callreturn = yes

callgroup = 1

pickupgroup = 1

[xxxx]

fullname = xxxx

email = xxx@xxx.xxr

secret = xxx

zapchan = 1

hasvoicemail = no

vmsecret = xxx

hassip = yes

hasiax = no

hash323 = no

hasmanager = no

callwaiting = no

context = international
```

Fig 7 */etc/asterisk/user.conf*

4.5 Équations

Nous déduisons donc de cette expérience que la limite de nombre de clients d'un serveur astérisque est proportionnelle à sa capacité de la mémoire. Ainsi, nous avons le nombre de clients maximum désigne par maxicli

$$maxcli=RAM/240k \qquad (1)$$

Conclusion

Ces analyses nous permettent de voir de plus prés ce qui se passe dans le serveur et surtout au niveau de la consommation de la ressource et des éventuels débordements de la mémoire du serveur. Dans notre cas on a utilisé le protocole Sip mais d'autres protocoles peuvent avoir aussi des failles de sécurité au niveau de la matérielle.

La solution que nous avons proposée n'est pas une invention, mais seulement une application de ce qui existe déjà dans Debian et dans astérisque et qui se situe dans une liste de paramètres reconnus comme modifiable.

Une limitation de droit de chaque utilisateur de notre serveur nous mène donc a un serveur plus sécurise

5. Références

[1] Tanenbaum A.,« Réseaux-Architecture, protocoles, applications », InterEditions, 1996

[2] Hersnet O. Gurle D., « La voix sur IP », Dunod, 2005

[3] Varela M.,« Evaluation Pseudo–subjective de la Qualite´ des FluxVoIP », irsa-INRIA, 2005

[4] Maiman M., « Télécoms et réseaux », Masson, 1997

[5] Battu D., « Télécommunications-Principes, infrastructures et services », Dunod, 2002

[6] http://www.asterisk.org

[7] http://www.debian.org

[8] Faggion N., « Le GPRS-Du Wap à l'UMTS »,Dunod 2002

[9] Pugolle G., « Les réseaux », Eyrolles, 2003

[10] omer D.E., « TCP/IP », Eyrolles,2004

[11] Lohier S., Présent D., « Transmissions et réseaux », Dunod, 2007

[12] Géron A., « WI-FI », Dunod, 2006

[13] Macchi C., Guilbert J.F., « Téléinformatique », Dunod, 1988

[14] Meggelen D.J.V. « Asterisk » O'reilly 2007

[15] Youngmi J., «TCP/IP traffic dynamics and network performance», Stanford University , 2001

[16] Jean-Marie G., Le Ha D., « Modélisation analytique du protocole TCP », LAAS-CNRS ,2000

[17] Y-L.CHEN and B-S CHEN. « Model-base multirate representation of speech signals and its application to recovery of missing speech packets », IEEE Trans.Speech Audio Processing, Mai 1997

[18] Dudet M., Collet P.,and Hersent O., «Téléphonie sur internet : quelles perspectives? », Les services de l'internet, p11, Juin 1998

[19] International Télécommunication Union(UIT), « Appendice i:Algorithme simple de haute qualité pour le masquage des pertes de paquet en codage g711 », Recommandation UIT-T G.711 – Appendice I, septembre 1999

[20] http://www.digium.org

[21] http://www.wikipedia.org