misc

MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

France Metro : 7.45 Eur - 12.5 CHF
BEL. LUX. PORT.CONT : 8.5 Eur - CAN : 13 $C
MAR : 75 DH

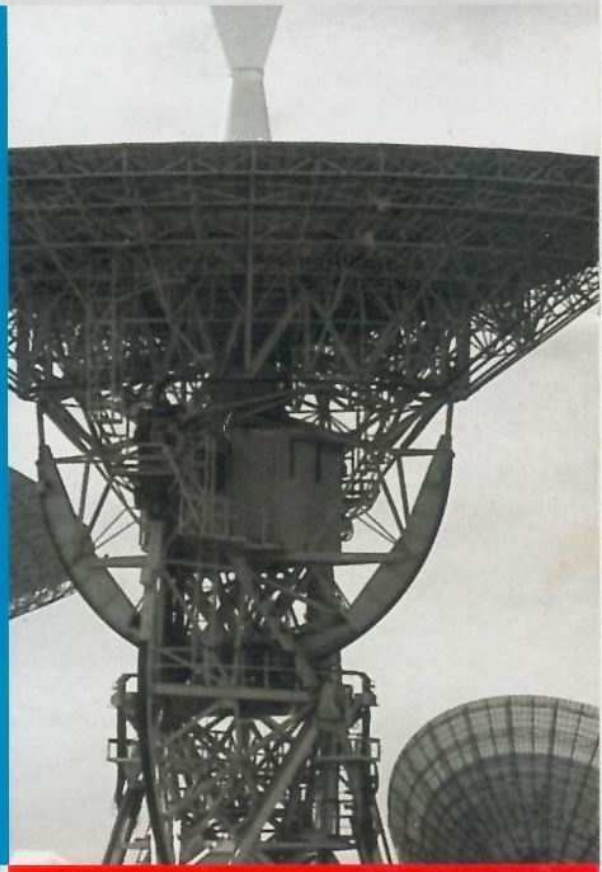
16 novembre décembre 2004

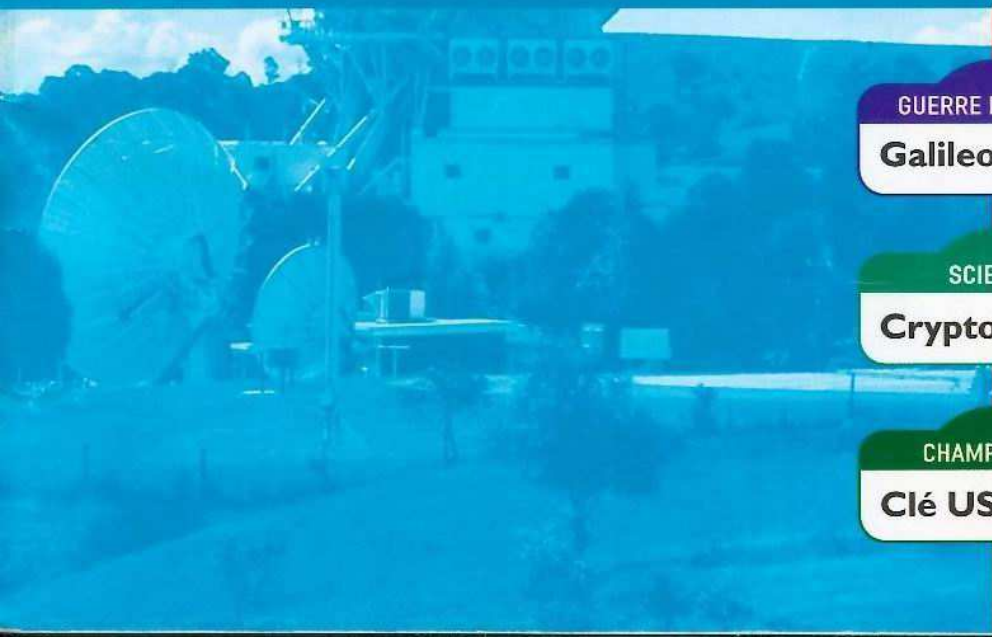
100 % SÉCURITÉ INFORMATIQUE

Télécoms

Les risques des infrastructures

Téléphonie fixe/mobile/VoIP





GUERRE DE L'INFO

Galileo, concurrent Européen du GPS

SCIENCE

Crypto : le chiffrement à flot

CHAMP LIBRE

Clé USB : la nouvelle menace

Sommaire

[1] http://www.01net.com/article/251625.html

[2] http://www.sstic.org

Édito

Faites de l'indépendance !

Je viens d'avoir la chance de passer 15 jours à Ouagadougou, au Burkina Faso, et je dois reconnaître que ce séjour fut, en dehors de quelques soucis gastriques, extrêmement enrichissant. La Flag du soir (les connaisseurs apprécieront, hein Monique ;-) ne compensa nullement le manque de ma moitié (oui, oui, c'est bien un message personnel pour me faire pardonner cette longue absence et éviter de me faire défenestrer à mon retour ;-) Merci à tous pour votre accueil !

Quoiqu'il en soit, j'ai découvert ici un sens encore plus fort pour le thème « d'indépendance technologique ».

En Europe, on commence à prendre conscience du risque de dépendre d'une unique source d'approvisionnement, que ce soit pour des systèmes d'exploitation ou des routeurs, par exemple. Les enjeux sont certes économiques, mais aussi stratégiques. Des projets, comme celui d'un Linux sécurisé certifié EAL5 [1], montrent que l'État a bien pris ce risque en considération, même si la réalisation concrète suscite de nombreuses questions sur les forums et autres listes de discussion. Idem avec le système européen Galileo par opposition au GPS américain (cf. article dans ce même numéro, d'ailleurs).

En Afrique, un éditeur de système d'exploitation ou un fabriquant de routeurs (exemples toujours pris au hasard bien sûr ;-) s'en donnent à cœur joie : ils forment les personnels dès le plus jeune âge dans les écoles et universités, ou fournissent logiciels et matériels, souvent gratuitement... la première année. Mais ces pays ont-ils réellement les moyens de payer ?

Doivent-ils pour autant être exclus de la révolution des Technologies de l'Information et Communications qui s'opère actuellement ? Les discours des politiques africains montrent bien qu'ils ont compris les enjeux que représentent pour eux cette révolution, et tentent de se donner les moyens de suivre cette évolution. Et malgré un manque de moyens flagrant, les gens font preuve d'une débrouillardise et d'une efficacité proportionnellement hallucinante.

Un autre problème qui se pose concerne la formation : il sort des écoles et universités trop peu d'ingénieurs informaticiens par pays et par an. Dans le même temps, les pays accèdent à beaucoup de ressources informatiques (systèmes d'exploitation, ordinateurs, routeurs...), mais qui pour les administrer correctement ? On voit déjà avec les pays de l'Est de l'Europe ce que ça donne : des moyens informatiques importants, mais un manque de personnel pour s'en occuper, ce qui transforme la région en véritable paradis pour les pirates.

Quel rapport avec la sécurité me direz-vous ? L'uniformisation des systèmes est un facteur important d'insécurité : imaginez un monde rempli des mêmes systèmes d'exploitation ou des mêmes routeurs déployés à travers tout un continent, et administrés par des personnes ne les maîtrisant pas ou peu... On dit souvent que la richesse vient de la diversité, mais la sécurité aussi dans une certaine mesure. Et pour atteindre cela, il est nécessaire de disposer de sa propre indépendance, tant au niveau des moyens que des connaissances.

Enfin, je ne pouvais pas conclure cet édito sans toucher quelques mots de l'événement maintenant incontournable, exceptionnel, que dis-je, extraordinaire qui se déroulera encore une fois à Rennes, les 1-2-3 Juin 2005 : le Symposium sur la Sécurité des Technologies de l'Information de la Communication (SSTIC). Nous avons déjà lancé l'appel à contribution, que vous pouvez consulter dans son intégralité sur le site [2]. La thématique retenue est « la lutte informatique » :

- → le contexte, c'est-à-dire les aspects politiques, juridiques, culturels, économiques et stratégiques ;
- → les techniques, outils et méthodes associés, qui sont au cœur des problèmes de sécurité informatique ;
- → les questions de post-intrusion, au travers l'analyse post-mortem par exemple.

Bien évidemment, ceci n'est nullement une incitation à contribuer ou participer... quoique ;-)

Bonne lecture,

Frédéric Raynal

Le projet de navigation par satellite européen : Galileo

« C'est un projet important pour l'avenir de l'Europe et son affirmation comme entité politique autonome et maîtresse de son devenir en toutes circonstances. Nous ne saurions en effet accepter une vassalisation de l'Europe en matière spatiale. »
Le président Jacques Chirac.

En 1994, l'Union Européenne et l'ESA (European Space Agency) lancent le programme Galileo, afin d'offrir une alternative au système GPS. Ce système de navigation par satellite, développé par le Ministère américain de la Défense à des fins initialement strictement militaires (cf. Arpanet), a évolué vers des activités civiles comme les transports et la géolocalisation. Seul système existant, le GPS était alors en position de monopole sur un marché économiquement rentable avec un atout stratégique indéniable (contrôle total).

Le projet « GALILEO » repose sur une constellation de trente satellites et des stations terrestres permettant de fournir des informations concernant leur positionnement à des usagers de nombreux secteurs tels que le transport (fret aérien, recherche d'itinéraire, localisation de flottes véhicules, systèmes de guidage, etc.), les services sociaux (par exemple aide aux handicapés ou aux personnes âgées), la justice et les douanes (contrôles frontaliers, surveillance des détenus), les travaux publics (systèmes d'information géographique), le sauvetage de personnes en détresse ou les loisirs (orientation en mer et en montagne, etc.) et bien sûr des applications militaires.

Il existe aujourd'hui deux réseaux de navigation par satellite : le GPS (USA) et Glonass (Russie), développés à l'origine par et pour les militaires (cf. développement d'Arpanet). *A contrario* le projet européen sera exploité par des partenaires privés avec éventuellement des applications militaires : une différence notable. (Le système russe n'ayant pas suscité de véritables applications civiles, le seul vrai concurrent du GPS est, à l'heure actuelle, européen).

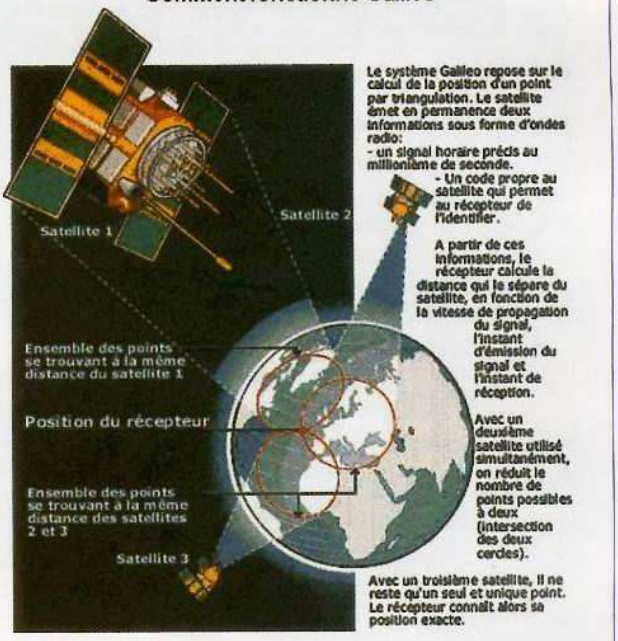
Ainsi, pour mettre en place les financements privés pour les phases de déploiement et d'exploitation et, à terme, devenir l'exploitant de Galileo, trois consortiums étaient à l'origine en compétition pour obtenir la concession.

Avant la fin de l'année 2004, la société commune constituée par la Commission européenne et l'Agence Spatiale européenne (*Galileo Joint Undertaking*) [1] devra choisir un des deux consortiums en lice après l'exclusion d'OHB et le retrait d'Eutelsat au début du mois de Septembre.

→ Le premier, iNavSat, réunit le groupe européen d'aéronautique, d'espace et de défense EADS, le groupe français d'électronique militaire Thales et les opérateurs de satellites Inmarsat, britannique, et SES Global, luxembourgeois.

Figure 1

Comment fonctionne Galileo



→ Le second, rebaptisé récemment Eurely, est composé du groupe français de BTPVinci Concessions, de l'entreprise française de télécommunications Alcatel, du groupe italien Finmeccanica ainsi que de Capgemini et SFR.

1. Les pressions « diplomatiques » américaines

« Galileo est pratiquement mort », lâchait le 17 janvier 2002 Gilles Gantelet, porte-parole de la commissaire européenne chargée des transports, au quotidien en ligne *Wired News*. À l'époque, la Commission européenne se dit exaspérée par la campagne de dénigrement orchestrée par les États-Unis à l'encontre du projet européen Galileo.

Sur ces thèmes, la validation du projet Galileo et son financement ont donné lieu à une lutte d'influence entre l'administration américaine et la Commission européenne pour gagner à leur cause respective les gouvernements de l'UE, durant la phase de validation du projet et sur les aspects techniques une fois le projet voté.

Une volonté délibérée de faire échouer les négociations entre pays européens participant au programme était déjà exprimée en 1992 dans un rapport du *Space Policy Advisory Board* [2] : « *Si les États-*

Marc Brassier
webmaster@guerreco.com

Pour mieux comprendre les différences entre les systèmes Galileo et GPS, un tableau récapitulatif :

	GALILEO	GPS
Satellites	30 satellites (environ) dont : → 25 en orbite moyenne (20 00 km) → 5 satellites géostationnaires servant à détecter les dysfonctionnements (à 36 000 km)	24 satellites à 20 000 km
Orbites	→ Inclinée à 56° par rapport à l'équateur → 3 plans → Une révolution en 14 h	→ Inclinée à 60° par rapport à l'équateur → 6 plans → Une révolution en 12 h
Couverture	Terre entière	Mauvaise aux latitudes élevées
Précision	3 à 4 m	5 à 10 m
Coût	3,2 à 3,4 Milliards d'euros (soit l'équivalent du tunnel TGV Lyon-Turin ou de 150 km d'autoroutes semi-urbaines)	7,5 Milliards $ (pour la modernisation et le passage au GPS III)
Bénéfices	→ 10 Milliards d'euros d'ici 2010 → Équilibre budgétaire prévu en 2015	40 Milliards $ (estimation 2005)
Propriétaires	Société internationale d'économie mixte composée de : → l'Union Européenne → l'ESA (Agence Spatiale Européenne) → la Banque Centrale Européenne → toute entreprise privée (contre un paiement de 5 Millions d'euros)	Département de la Défense Américain (Pentagone)
Avantages	→ Précision supérieure et constante → Service intègre → Couverture de toute l'Europe → Projet dirigé par une équipe internationnale	→ Satellites multi-fonctions, capables de détecter les explosions nucléaires par exemple
Défauts	(aucun répertorié actuellement : état de projet.)	→ Origine Militaire → Fiabilité médiocre et précision variable

Unis continuent à fournir un signal gratuit de haute qualité, il est douteux que quiconque veuille engager les dépenses nécessaires pour bâtir un système spatial global comparable. Le contrôle des États-Unis sur le segment spatial de GPS leur permet de protéger leurs intérêts militaires, cependant que la compétition commerciale sur les équipements terriens tend à promouvoir la croissance économique globale » ; ou bien encore cet extrait issu des travaux de la Rand corporation (un *think thank* Américain) : *« Décourager la prolifération des systèmes concurrents et donner à l'industrie des États-Unis la meilleure chance de maintenir son leadership actuel dans un marché commercial croissant »* [3]

2. Comment les États-Unis ont essayé d'influencer la politique spatiale européenne ?

Tout débute le 20 novembre 2001, ce jour-là est présenté un rapport de PriceWaterHouseCooper commandé par la commission européenne qui valide la rentabilité économique de Galileo en estimant que le projet offrira un bon ratio coût-bénéfice, de l'ordre de 4,6. Aucun projet d'infrastructure européen n'atteint un tel ratio.

Cette expertise démontre l'intérêt financier du projet, estimant les bénéfices à 17,8 milliards d'euros pour un coût de 3,2 milliards d'euros. Sur la période 2005-2023, d'autres analyses avancent que le bénéfice induit par Galileo pourrait atteindre 80 milliards d'euros. Toujours selon les mêmes études, Galileo créerait 140 000 emplois en Europe.

À partir de ce document qui viabilise un « GPS » européen, les Américains, qui ne souhaitent pas que leur monopole soit remis en cause, tenteront de s'opposer à la réalisation du projet.

3. Les Européens ont ils raison de se réjouir ?

Le lancement du projet ayant été validé, il restait les questions liées à l'interopérabilité des deux systèmes pouvant satisfaire les deux parties.

Presque deux ans après l'accord de l'UE sur le développement de Galileo, et après deux jours de négociation à Bruxelles, les 24 et 25 février 2004, Les États-Unis et l'UE sont parvenus à un accord sur la coopération entre les systèmes de navigation par satellite Galileo et GPS. Les deux parties ont annoncé avoir réglé la plupart des points restés en suspens jusqu'ici, dont la question la plus difficile de la fréquence qui sera utilisée par le futur signal ouvert (grand public) de Galileo.

Un point sur les concessions Européenne et sur... l'inflexibilité Américaine

Possibilité de brouillage du signal de Galileo

Les fréquences finalement choisies permettront aux États-Unis de brouiller le signal Galileo dans des zones de conflit, sans dégrader pour autant le signal militaire du GPS. Mais, même si l'UE a cédé du terrain sur la question des fréquences, la Commission affirme que la qualité du signal ne s'en ressentira pas. Côté américain, on admet avoir eu « quelques réserves » vis-à-vis de Galileo dans le passé mais on se dit maintenant « enthousiaste ». « Aujourd'hui, nous nous rendons compte qu'il y a des avantages mutuels », a déclaré Ralph Braibanti, du Département d'État américain.

Risque d'interférence entre les fréquences de Galileo et les fréquences militaires

Pour rappel, le principal problème qui restait à régler ces derniers mois concernait les **risques d'interférence** entre les fréquences de Galileo et les fréquences militaires – ce qu'on appelle le « code M » (signal militaire protégé) – du GPS américain. Une première avancée avait été enregistrée en novembre 2003, quand les Européens avaient proposé de modifier la modulation des signaux PRS (*Public regulated Service*) de Galileo, cryptés et essentiellement réservés aux besoins des institutions publiques. Cela faisait alors deux ans que les États-Unis dénonçaient l'utilisation des mêmes fréquences par le PRS de Galileo et le signal militaire du GPS, qui ne permet pas de brouiller sélectivement l'un des deux signaux sans dégrader l'autre.

Une solution du même type devait encore être trouvée pour les signaux publics de Galileo, ce à quoi sont parvenus les négociateurs à Bruxelles ces 24 et 25 février 2004.

Washington insistait auprès des européens sur la nécessité de pouvoir brouiller ces signaux pour des raisons de sécurité, par exemple, en cas de guerre. Or, l'Union internationale des télécommunications – un organisme de l'ONU installé à Genève – avait attribué à Galileo des fréquences qui chevauchaient celles du GPS. En acceptant de décaler légèrement les fréquences de Galileo, les européens laissent aux américains la possibilité de brouiller localement des signaux civils, sans altérer leur propre système militaire. Une concession importante et discrète, qui a ouvert la porte à l'accord transatlantique.

Les américains peuvent brouiller tous les signaux, l'Europe n'a pas encore cette capacité

L'accord est présenté comme « excellent pour les deux parties. Le signal ouvert sera aussi performant que ce qui était prévu au départ », a déclaré devant la presse Heinz Hilbrecht, directeur à la Commission européenne, précisant aussi que les fréquences choisies offriront « une qualité suffisante pour le PRS ». Le directeur de l'Office américain de l'Espace et de la Technologie, Ralph Braibanti, se félicitait que « la structure des signaux de Galileo est telle qu'elle ne limitera pas les capacités de guerre de notre gouvernement ».

« Dans une zone en crise, où des combats se déroulent, les forces de l'OTAN auront la faculté de refuser l'accès à tous les signaux, y compris le GPS et Galileo » a-t-il expliqué. Concrètement, les USA auront la possibilité de brouiller les signaux Galileo, qui pourraient être utilisés à des fins hostiles, sans dégrader leur propre signal militaire. Heinz Hilbrecht a aussi indiqué que l'UE allait se pencher sur la « possibilité de protéger le PRS ». [7]

Interopérabilité du système GPS et Galileo sur les récepteurs

La structure des fréquences retenues permettra aux détenteurs d'un seul récepteur de recevoir des signaux Galileo et GPS, même si la constellation de départ est légèrement différente. Comme l'accord en prévoit la possibilité, M. Hilbrecht a indiqué que les experts européens avaient l'intention d'optimiser le signal retenu (dit « Boc 1,1 ») pour améliorer encore les performances de Galileo. Heinz Hilbrecht parle d'un signal « classique » pour la future génération de GPS (GPS III) et « optimisé » pour Galileo (en estimant que ce dernier deviendra la norme au fil du temps) mais tout à fait interopérables. Les propriétaires actuels d'un récepteur GPS ne devront pas se procurer un nouveau récepteur pour recevoir les signaux du GPS nouvelle formule, mais les nouveaux appareils offriront des signaux de meilleure qualité. Selon Ralph Braibanti, le remplacement des satellites actuels du GPS par des satellites qui intégreront le signal commun se fera entre 2010 et 2020.

Une attitude européenne bien timide

La commission européenne a accepté successivement de déplacer la bande de fréquence du GPS, puis d'accorder aux USA un droit de brouillage régional du GPS et de Galileo. Si l'objectif de Galileo était de doter l'Europe d'un outil d'indépendance technologique et militaire, on peut s'interroger sur le sens des concessions faites sous la pression américaine. Galileo ne se présente plus aujourd'hui que comme un concurrent des seules applications civiles du GPS !

Lors de la Guerre du Golfe, les Américains n'ont pas hésité à brouiller le signal GPS au-dessus d'Israël afin d'interdire toute riposte

aux attaques de Scuds irakiens. Quelles seront les critères déterminants le brouillage des fréquences de Galileo et n'est-ce pas cette question qu'il fallait poser avant d'entamer les négociations ? Heinz Hilbrecht a d'ailleurs déclaré à l'issue de la signature de l'accord : « Nous allons discuter afin de prévoir des situations dans lesquelles on ne brouille pas le PRS ». [8]

Quant au passage de la norme BOC 2.2 à BOC 1.1, les ingénieurs européens ont trouvé le moyen de donner à la norme BOC 1.1 une qualité de précision égale à celle du BOC 2.2. L'accord engage les Américains à utiliser cette norme BOC 1.1 pour leur GPS III. Mais un « proche du dossier » dans un article de *La Tribune* datée du 1er mars 2004 semblait légèrement pessimiste : « Nous avons toutefois conservé la possibilité technique de revenir sur BOC 2.2 si, d'aventure, les Américains ne tenaient pas leur promesse ». Clairement l'Union européenne, avec 6 ans d'avance sur son concurrent, s'adapte à l'infériorité technique du système GPS mais assumerait les charges liées à un retour à BOC 2.2 si les Américains changeaient d'avis.

Un retard technologique qui s'est déjà ressenti lors d'une démonstration au Printemps 2000 devant les autorités grecques, des chars français n'ayant pas hésité à brouiller les transmissions GPS de leurs concurrents commerciaux anglais et américains... « *preuve* » de la vulnérabilité du GPS [8].

Une vulnérabilité du GPS qui serait l'une des leçons de la guerre en Irak, selon un article de la revue *Scientific American*. Les Irakiens se seraient en effet portés acquéreurs de *« brouilleurs » rapidement mis hors service car ils utilisaient également un système GPS facilement repérable* fournis par Aviaconversiya, une société basée à Moscou qui développe des produits permettant de brouiller les systèmes radio, y compris le GPS employé dans les avions et les bombes. (Moscou aurait démenti toute implication... les brouilleurs étant inclus dans les accords concernant l'embargo sur l'Irak).

Des brouilleurs dont on peut trouver le mode d'emploi sur plusieurs e-zines [11], ce qui aurait sûrement intéressé en Mai 1996 le « rebelle » Tchéchène Doudaiev dont on se souvient du sort que lui ont réservé les forces russes (lancement d'un missile), car il avait eu le malheur d'utiliser un téléphone satellite, lui aussi équipé d'un GPS... [12]

Sur ces questions, et pour une utilisation militaire, il est indispensable que Galileo diffuse un signal spécial pouvant être partagé avec

Petit agenda des relations États-Unis / Europe de 2001 à 2004

1er décembre 2001

Dans une lettre, Paul Wolfovitz demande aux 15 ministres de la Défense européens de reconsidérer le projet Galileo, au vu des risques d'interférence avec le GPS. Car les fréquences de transmission qu'utilisera Galileo sont dans le même spectre que celles des prochaines mises à jour du système américain GPS vers sa version 3 (prévu vers 2015).

Ainsi, selon un extrait de cette lettre adressée par Paul Wolfowitz « *The addition of Galileo services in the same spectra will significantly complicate our ability to ensure availability of critical GPS services in times of crisis or conflict and at the same time assure that adversary forces are denied similar capabilities.* »[4][6]

4 décembre 2001

Conséquence immédiate de la lettre du N°2 du pentagone, les ministres des Finances du Royaume-Uni, de l'Allemagne, du Danemark, de l'Irlande, de l'Autriche et de la Suède indiquent « *ne pas être en mesure de donner leur avis sur la poursuite du financement du projet Galileo* ».

15 décembre 2001

Conseil européen de Laeken à l'ordre du jour, deux sujets sur lesquels les USA et l'UE sont en désaccord : le brevet européen et le projet Galileo. Les pro-galileo réaffirment tout de même « l'importance stratégique », et la France et l'Espagne soulignent que le rapport coûts-bénéfices sera positif, des profits devant même être dégagés à partir de 2011 (La rentabilité à terme devrait être de 17,8 milliards d'euros pour un investissement de 3,9 milliards d'euros).

18 janvier 2002

Publication d'un contre-argumentaire par la Commission européenne : « *Galileo, un impératif pour l'Europe* ». Ce document répond point par point aux arguments avancés par la diplomatie américaine depuis le mois de décembre sur les enjeux financiers, politiques et économiques.

Début février 2002

Vidéoconférence entre les Espagnols (présidence de l'Union), fervents défenseur du projet, et deux hauts fonctionnaires américains qui critiquent le coût du système.

8 mars 2002

Conférence de presse à Washington, pour expliquer la position américaine ? ou convaincre l'opinion publique ? Citation : « *Les États-Unis ne voient pas la nécessité du projet Galileo dans la mesure où le GPS pourra satisfaire les besoins de tous les utilisateurs dans le monde dans un futur prévisible* ».

Serge Plattard, directeur des relations internationales du CNES, se souvient que les États-Unis ont tenu un discours comparable au moment de la mise sur pied du lanceur Ariane. « *Dans les années 1970, les Américains nous avaient expliqué que c'était inutile de développer la fusée Ariane car ils pouvaient lancer des satellites pour nous, raconte-t-il. Or, quand on a voulu lancer des satellites commerciaux, notamment les satellites de télécommunications franco-allemands Symphonies, ils ont exigé des appels d'offres et l'attribution au meilleur offrant, peu importe qu'il soit américain ou européen. Les Européens ont alors décidé de créer leur propre lanceur.* » [12]

9 mars 2002

Envoi par le gouvernement américain d'un dernier communiqué sur Galileo aux gouvernements européens réitérant l'inquiétude des Américains en ce qui concerne les interférences entre signaux et le recours aux fonds privés pour financer Galileo.

11 mars 2002

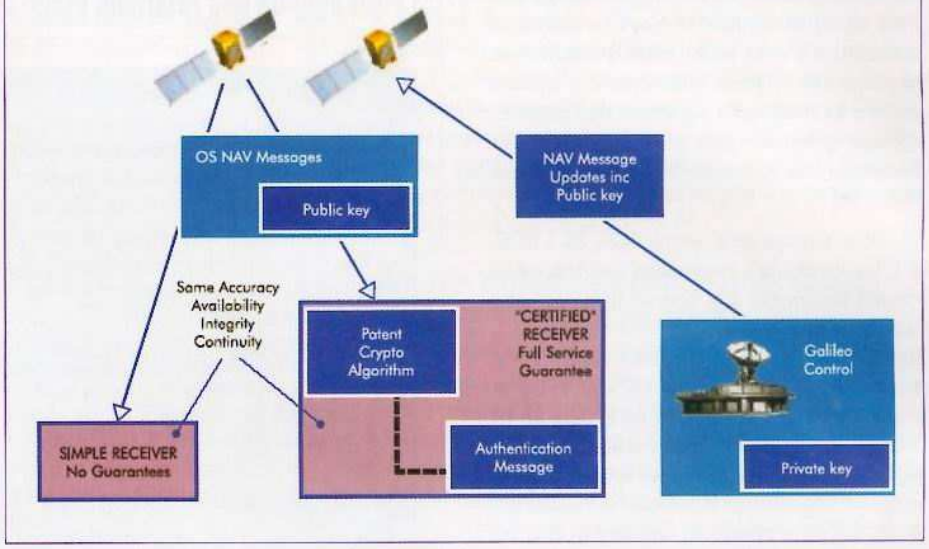
La Commission européenne publie une nouvelle réponse plus directe et moins consensuelle à l'argumentaire américain. « *Les États-Unis font preuve d'une étonnante sollicitude en « prévenant » leurs amis européens de la non-rentabilité, à les entendre, de Galileo...* » ou « *les Européens n'entendent pas adopter à leur tour une mentalité protectionniste et monopolistique. Ils ne nient pas l'intérêt du GPS en dépit de la supériorité de Galileo...* » [5]

26 mars 2002

Validation de Galileo par le Conseil européen des Transports.

d'autres utilisateurs gouvernementaux (police, sécurité civile). Par ailleurs, le signal doit être sécurisé c'est-à-dire chiffré et protégé contre un certain niveau de brouillage offensif pour un emploi dans des conditions dégradées, notamment en opérations. Enfin, le système doit être compatible avec les systèmes militaires existants ou en cours de développement (radars, MIDS/JTIDS). La défense française s'est déclarée prête à participer au développement d'un signal gouvernemental à condition que quatre pays au minimum y soient associés. Le coût de ce signal est estimé à 150 millions d'euros à répartir sur huit ans, mais à l'heure actuelle, aucun financement n'est prévu. A priori des questions qui seront gérées dans le futur par le discret GSSB (*Galileo Security System Board*).

Figure 2



« *Galileo offrira donc sûrement un service sécurisé à l'aide de solutions de cryptographie à certains utilisateurs triés sur le volet, qui auront été contrôlés, authentifiés et répertoriés avec la possibilité de tracer et dater chaque information* », souligne Bernard Mathieu du CNES [14]. Mais l'on pourrait admettre que les Européens adoptent eux aussi le circuit intégré SAASM *(selective available and anti-spoofing module)*, fourniture cryptographique du GPS de nouvelle génération (GPS3) pour les appareils mobiles.

Mais....

L'entrée du fonds américain Texas Pacific Group (TPG) dans le capital de l'opérateur de satellites Eutelsat avait ému certains analystes soucieux de la souveraineté européenne, notamment au regard du contrôle du futur système de GPS européen Galileo.

Les fonds d'investissement anglo-saxons témoignent d'un intérêt croissant pour le domaine du Spatial européen. Depuis mars 2001, GE Capital détient 30% du capital et près de 25% des droits de vote de l'opérateur européen de satellites SES Global, né de la fusion de sa filiale GE Americom et de SES Astra. Candidat malheureux à l'entrée dans le capital d'Eutelsat fin 2002-début 2003, Carlyle Group a, quelques mois plus tard, pris le contrôle de Fiat Avio, fournisseur de composants des fusées Ariane, et une participation importante (et qui devrait augmenter encore) dans le Britannique QinetiQ, présent notamment sur le segment stratégique des technologies pour microsatellites. APAX Partners et Permira, quant à eux, sont sortis vainqueurs de la vente des actions d'Inmarsat mises aux enchères fin 2003. A travers le consortium iNavSat, Inmarsat est lui aussi, tout comme Eutelsat l'était, candidat à la gestion de la concession Galileo...

Un peu de prospective

Les RFID, la monétique ainsi que le Wi-Fi, autant d'applications pouvant converger avec des systèmes de localisation par satellite. On peut prévoir que les retombées économiques futures engendreront une concurrence acharnée. Les querelles actuelles sur les normes GSM européennes et Américaines CDMA (qui incluent un systèmes GPS) « *conforme à la loi CALEA* » (*Communications Assistance for Law Enforcement Act*), qui permet aux forces de l'ordre et services de renseignements américains d'accéder à toute communication effectuées par ce biais en sont déjà un bon exemple.

Références

[1] **http://www.galileoju.com/**

[2] **http://www.senat.fr/rap/r00-293/r00-29332.html :** rapport du sénat sur la politique spatiale européenne

[3] La Rand Corporation : **http://www.rand.org/**

http://www.wired.com/news/conflict/0,2100,47739,00.html : Afghanistan coupure GPS

[4] Extrait de la lettre de Paul Wolfovitz sur Herald Tribune : **http://www.iht.com/articles/42335.htm**, et une copie de la lettre sur Zdnet.fr : **http://images.zdnet.fr/img/zdnn/2002/lettreusa3.jpg**

[5] **http://europa.eu.int/comm/dgs/energy_transport/galileo/doc/galileo_info_note_2002_03_26_fr.pdf**

[6] Voir les articles de M. Guillemin, Zdnet : **http://www.zdnet.fr/actualites/technologie/0,39020809,2106212,00.htm**

[7] Reuters: **http://www.reuters.fr**

[8] **http://www.infoguerre.com** : « cécité stratégique » de Ronan Jegou, **http://www.infoguerre.com/article.php?op=Print&sid=747**

[9] Transfert.net : **http://www.transfert.net/a8676**, *Le talon d'Achille de l'armée américaine*, par Jean-Marc Manach

[10] **http://www.sciam.com/print_version.cfm?articleID=00079DD3-DAA0-1E96-8EA5809EC5880000**

[11] **http://www.gbppr.dyndns.org/PROJ/mil/gps/ :** conception d'un brouilleur GPS

[12] **http://www.acat.asso.fr/campagne/** DossierTCHECHENIE.rtf : Mort de Doudaiev

[13] Rapport Galilei : **http://europa.eu.int/comm/dgs/energy_transport/galileo/doc/**

[14] **http://privilege.ledevoir.com/2002/05/25/1830.html** : Pauline Gravel, *La guerre de l'espace*

La clé USB : votre nouvel ennemi

L'objectif de cet article est d'étudier les risques et de sécuriser l'emploi des clés USB en milieu professionnel. En effet, depuis 3 ans, le port USB est utilisé pour connecter des systèmes de stockage de masse utilisant la technologie des mémoires flash (non volatiles et réinscriptibles) sous la forme de clés. Ce système peu encombrant, de capacité de stockage importante et d'usage grand public a détrôné cette année le lecteur de disquette. Mais l'utilité d'une clé USB n'est pas limitée au stockage des données : certains utilitaires permettent de bloquer l'accès à votre PC ou à certains dossiers, de récupérer et lire vos emails à partir de n'importe quel PC connecté au Net, ou encore rendent le contenu de votre clé inaccessible. Nous pouvons alors imaginer ce qu'un programme malveillant peut obtenir comme information à votre insu. Il est peut être déjà trop tard...

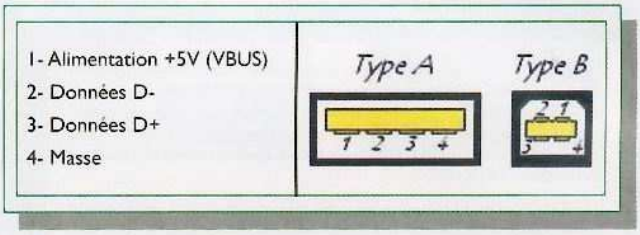
Quelques rappels sur la clé USB

La norme USB

La norme USB (*Universal Serial Bus*) est née de l'alliance en 1994 de sept grands groupes de l'informatique : Microsoft (le leader), IBM, Intel, Compaq, DEC, NEC, et Northern Telecom. L'objectif était de simplifier la connexion des périphériques à un ordinateur. En janvier 1996, la norme USB 1.0 sort et les premiers produits apparaissent fin 1997. En 2000 sort la spécification 2.0. Cette nouvelle norme supporte toutes les caractéristiques de l'USB 1.1 : 1,5 Mbps (*Low Speed*), 12 Mbps (*Full Speed*), rajoute une vitesse de 480 Mbps (*High Speed*) et optimise l'utilisation de la bande passante.

Comme son nom l'indique, la norme USB permet sur un port unique et avec un système de câble universel de connecter tous types de périphériques : clés de mémoire, disques durs, scanners, modems ADSL, imprimantes, souris, claviers, joysticks, webcams, etc. La norme USB permet le « *hot Plug and Play* » et le « *hot-swapping* », c'est-à-dire que l'on peut utiliser le nouveau périphérique et le débrancher sans redémarrer l'ordinateur. Elle permet d'alimenter en électricité les périphériques jusqu'à 500 mA et 5V. Cela évite la profusion de prises électriques et réduit le coût de production du périphérique.

Les drivers sont prédéfinis dans les systèmes d'exploitation, il n'y a donc normalement pas besoin de charger le moindre driver avant d'utiliser le périphérique (pour Windows à partir de la version 2000). Il est possible de connecter 127 périphériques par port et d'éloigner un périphérique de seulement quelques mètres. La clé USB utilise une fiche appelée connecteur USB de type A (qui est utilisée également lorsque le périphérique se connecte sans câble) dont la forme est rectangulaire et sert généralement pour des périphériques peu gourmands en bande passante (clavier, souris, webcam). Les connecteurs dits de type B, de forme carrée, sont utilisés principalement pour des périphériques à haut débit (disques durs externes). L'architecture USB a pour caractéristique de fournir l'alimentation électrique aux périphériques qu'elle relie, en utilisant un câble composé de quatre fils : la masse, l'alimentation VBUS et deux fils de données appelés D- et D+.



Le Protocole USB

Un hub USB

La norme USB permet le chaînage des périphériques, grâce à une topologie en bus ou en étoile. Les périphériques sont soit connectés les uns à la suite des autres, soit ramifiés. La ramification se fait à l'aide de boîtiers appelés hubs, comportant une seule entrée et plusieurs sorties. Certains sont actifs (fournissant de l'énergie électrique), d'autres passifs. Le bus USB utilise un protocole maître/esclaves où le maître est l'hôte et où les esclaves sont les périphériques. Ce protocole est construit sur le principe de l'anneau à jeton (*token ring*). Lorsque l'hôte désire communiquer avec un périphérique, il émet un jeton (un paquet de données, contenant l'adresse du périphérique, codée sur 7 bits) qui lui est destiné. Si un périphérique reconnaît son adresse dans le jeton, il envoie un paquet de données en réponse. Sinon, il fait suivre le paquet aux autres périphériques connectés au bus.

La trame USB, appelée transaction, est constituée de paquets juxtaposés :

- → *Token* (ou jeton) : indique le type de la transaction qui va suivre et a pour but de transporter l'adresse USB et le sens du transfert (IN, OUT, SETUP) ;
- → *Data* : transporte les données utiles ;
- → *Handshake* : valide les données et rapporte les erreurs (ACK, NACK, STALL) ;
- → SOF (*Start Of Frame*) : indique le commencement d'une nouvelle trame.

Transaction USB

TOKEN Packet	DATA Packet	ACK Packet	SOF

Thierry Martineau
thierrymartineau@yahoo.fr

Paquet USB

Synchronisation	Packet ID	Packet Specific Information	CRC	EndOfPacket

Les différents types de transfert USB

	CONTROL TRANSFERT	ISOCHRONOUS TRANSFERT	INTERRUPT TRANSFERT	BULK TRANSFERT
Format des données imposé	oui	non	non	non
Taille maximale du bloc de données	Low speed : 8 octets Full speed : 64 octets	Full speed: 1023 octets	Low speed: 8 octets Full speed: 64 octets	Full speed : 64 octets
Accusé de réception, reprise sur erreur	oui	non	oui	oui

Il existe quatre types de transfert pour faire transiter des données entre l'hôte et un périphérique sur le bus (voir tableau ci-dessus) :

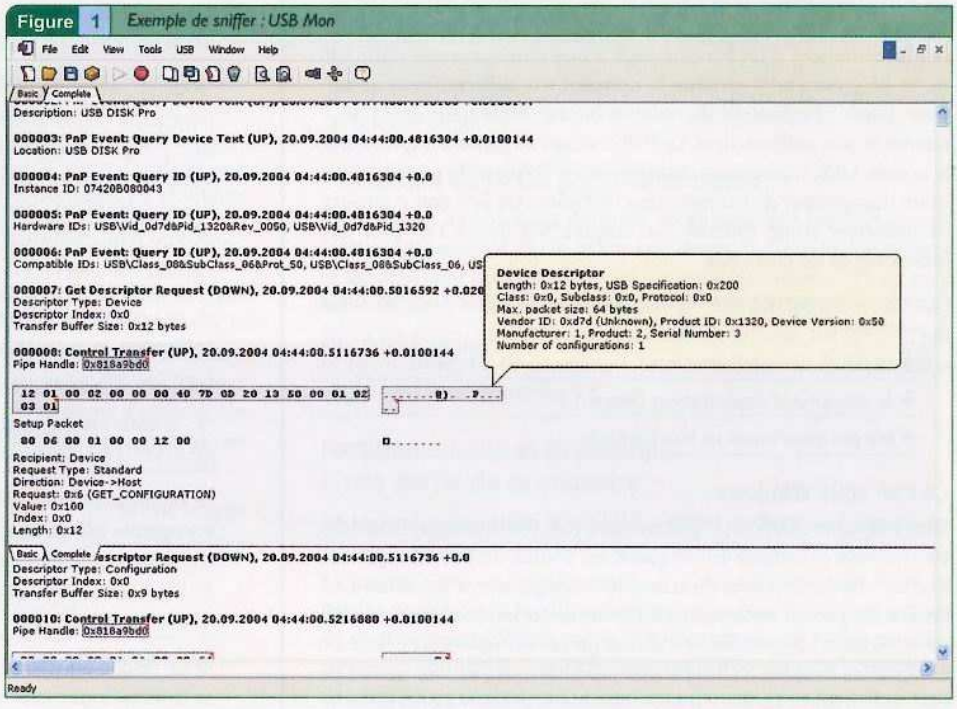
→ le transfert de commande (*control transfert*), utilisé pour les opérations d'initialisation et de configuration ;

→ le transfert d'interruption (*interrupt transfert*), destiné à des échanges limités en taille, pour les claviers USB par exemple ;

→ le transfert isochrone (*isochronous transfert*), utilisé pour des transferts nécessitant un flux régulier de données, comme par exemple les caméras ou les téléphones ;

→ le méga transfert (*Bulk transfert*), utilisé pour les gros transferts de données, pour les clés USB notamment.

Figure 1 *Exemple de sniffer : USB Mon*



L'énumération, le processus par lequel l'hôte identifie et configure le périphérique en lui donnant une adresse unique, le positionne dans la configuration qui lui a été donnée par les descripteurs, et charge le driver correspondant. Les descripteurs sont des blocs d'informations pré-formatés.

Tous les périphériques USB doivent obligatoirement posséder les descripteurs standards. Ils détaillent pour le compte de l'hôte des informations l'instruisant sur la nature de l'appareil (*Product ID*), sur son fabriquant (*Vendor ID* sur 16 bits), sur la version USB qu'il supporte, sur le nombre de manières dont il peut être configuré, etc. Cette phase d'énumération est totalement transparente et automatique pour l'utilisateur. C'est une gestion dynamique de la connexion et de la déconnexion des périphériques reliés à un bus USB.

Analyse de trame USB

Par analogie au monde des réseaux, nous utilisons un outil d'analyse de trames supportant le protocole USB. Ce logiciel s'installe sur le PC et intercepte tous les transferts des ports USB. Nous pouvons alors « écouter » les paquets échangés entre une clé USB et le

système d'exploitation lors de l'insertion de celle ci. Les logiciels SNOOPYPRO ou USB Monitor peuvent être utilisés par exemple. Nous observons (cf figure 1) l'identification de la clé USB (VID/PID) et la négociation des paramètres de fonctionnement. L'insertion d'une clé USB sur un ordinateur provoque environ une centaine de paquets USB.

Toutes les informations sont transmises en clair. Il suffit donc d'écouter la communication entre un ordinateur et un périphérique USB pour connaître le contenu des fichiers manipulés, mots de passe, etc. Nous verrons ultérieurement, toujours par analogie au réseau, la possibilité de perturber ce protocole USB en fabriquant des paquets « personnalisés » : générateur de paquets, scanner de périphériques connectés, etc.

La norme USB atteint son objectif initial de simplification de la connexion des périphériques à un ordinateur. Il n'existe pas nativement de préoccupation de sécurité, mais plutôt d'interopérabilité. L'étape suivante sera de s'affranchir de la contrainte du câble (connexion USB sans fil On-The-Go).

Les fonctionnalités possibles

Le PnP

Le « *Plug and play* » (PnP) est à la fois un concept et une norme développés principalement par Microsoft en 1995 visant à faciliter le raccordement d'un périphérique à une unité centrale hôte. On parle de « *hot PnP* » lorsque le concept est suffisamment abouti pour éviter à l'utilisateur de redémarrer son ordinateur après avoir connecté son périphérique. Le PnP, concept largement exploité par la norme USB, suppose un dialogue entre l'hôte et le périphérique et un changement de configuration de l'hôte que l'on doit connaître et maîtriser pour étudier les risques SSI liés à l'emploi des technologies de clés USB.

Les trois éléments suivants doivent répondre aux critères de la norme PnP :

- → le BIOS de l'ordinateur ;
- → le système d'exploitation (noyau) ;
- → les périphériques et leurs pilotes.

Le PnP sous Windows

Sous Windows 2000/XP, le gestionnaire PnP (*PnP manager*) maintient un contrôle centralisé en dirigeant les pilotes de bus afin que ces derniers fassent les énumérations, les configurations et mettent en œuvre les pilotes nécessaires à l'ajout et au fonctionnement d'un périphérique. Le gestionnaire d'énergie (*power manager*) prend en compte les besoins d'énergie des périphériques et leur alloue ce qui est demandé. Le Bus Driver contrôle la gestion d'énergie et le PnP.

Dans ce contexte, chaque périphérique énuméré est référencé par un pilote de bus. Ce pilote est construit selon la norme *Windows Driver Model* (WDM). Le gestionnaire PnP gère une arborescence de pilotes qui contient les informations des périphériques actifs dans le système. Il met à jour cette arborescence en fonction des mouvements de périphériques et des diverses allocations de ressources.

Lors du branchement d'une clé USB, les descripteurs transmettent le couple PID/VID à l'ordinateur hôte qui recherche alors dans tous ses répertoires un fichier `.inf` (comme information) contenant le couple PID/VID qu'il a reçu. Le fichier `.inf` indique quel driver charger. Dans le cas d'une clé USB connectée à un hub USB, lui-même connecté à un bus USB, trois fichiers INF seront sollicités : pour le bus, le hub et la clé. Celui qui permettra la connexion de la clé USB est le fichier `usbstor.inf`. Ce fichier `.inf` provoque le chargement du driver adapté qui est un fichier en `.sys` (`usbport.sys`, `usbhub.sys`, `usbstor.sys`).

Le logiciel *Regmon* dénombre près de 5 000 actions sur la base de registre lors du branchement d'une clé USB. Seules les actions les plus significatives sont présentées afin de faciliter la compréhension du mécanisme de prise en compte de la clé par le système d'exploitation.

À la connexion de la clé :

- → le système perçoit qu'un périphérique est connecté sur le bus USB. Il reçoit les identifiants du périphérique qu'il analyse dans la sous-clé :
  `HKLM\System\CurrentControlSet\Control\usbflags` ;
- → il procède ensuite à l'énumération dans la sous-clé :
  `HKLM\SYSTEM\CURRENTCONTROLSET\ENUM\USB` ;
- → il crée une clé correspondant aux identifiants fournis par le matériel connecté
  `HKLM\SYSTEM\CURRENTCONTROLSET\ENUM\USB\Vid_08ec&Pid_0011` ;
- → dans cette clé, il identifie et stocke les caractéristiques principales du périphérique. Grâce à ces paramètres, il détermine, par l'intermédiaire de `usbstor.inf`, le type et le rôle du périphérique et détermine le pilote nécessaire à son fonctionnement (dans le cas présent : `usbstor.sys`)
  `HKLM\SYSTEM\CURRENTCONTROLSET\ENUM\USB\Vid_08ec&Pid_0011\02169301 0A019FAA\Service\usbstor` ;
- → le système attribue des paramètres au périphérique : une référence, le service actif (`usbstor`) et une classe de matériel (`deviceclasses`) ;
- → la sous-clé est ensuite créée :
  `HKLM\SYSTEM\CURRENTCONTROLSET\ENUM\USBSTOR` ;
- → dans cette sous-clé on trouve l'ensemble des paramètres enregistrés afin de ne pas avoir à faire une énumération complète à chaque branchement de la clé USB
  `HKLM\SYSTEM\CURRENTCONTROLSET\ENUM\USBSTOR\Disk&Ven_IBM&Prod_Memory_Key&Rev_3.52` ;
- → un disque est affecté au périphérique pour permettre l'écriture et la lecture des informations qu'il contient ;
- → la lecture des paramètres spécifie aussi que le périphérique est amovible, ce qui entraîne la mise à jour des sous-clés `HKLM\SYSTEM\CURRENTCONTROLSET\ENUM\STORAGE\RemovableMedia` et `HKLM\System\CurrentControlSet\Control\DeviceClasses` ;
- → une lettre de lecteur est attribuée à la clé USB :
  `HKLM\System\MountedDevices\\DosDevices\E:` ;
- → des ressources (notamment l'énergie) sont allouées au périphérique selon la norme ACPI :
  `HKLM\SYSTEM\CURRENTCONTROLSET\ENUM\Root\ACPI_HAL\0000.`

Chaque action entraîne une mise à jour de toutes les autres sous-clés de la base de registre. À l'issue de toutes ces actions, la clé USB est accessible par l'utilisateur.

Le PnP sous Linux

Sous Linux, la gestion de l'USB est réalisée :

→ au niveau du noyau pour la configuration du bus USB (à partir du noyau 2.4.x) ;

→ au niveau applicatif pour le choix du pilote permettant la configuration dynamique de l'hôte et du périphérique lors de sa connexion physique.

Linux utilise l'interface UHCI (*Universal Host Controller Interface*) spécifiée par Intel et utilisée par les contrôleurs de la plupart des cartes mères. Lors du démarrage du système d'exploitation, le noyau de Linux charge automatiquement les modules nécessaires au fonctionnement du bus USB. Le pilote nécessaire à la connexion d'une clé USB est `usb-storage`. La commande `lsmod` permet de visualiser les modules et les pilotes :

```
[root@localhost root]# lsmod
Module Size Used by Not tainted
usb-storage 74624 0 (1)
scsi_mod 108968 1 [usb-storage] (3)
usb-uhci 26412 0 (unused)
usbcore 79392 1 [usb-storage hid usb-uhci] (2)
```

Le module `usb-storage` a été chargé (1). Le module `usbcore` spécifie au système qu'un périphérique de stockage est identifié sur le bus USB (2). Enfin le module `scsi_mod` a été chargé et fait le lien avec `usb-storage` (3). Il agit comme une passerelle entre l'USB et une émulation SCSI. De ce fait, tous les périphériques de stockage reconnus sur le bus USB deviendront des émulations de périphériques SCSI.

Le noyau Linux 2.6 apporte de réelles nouveautés au niveau de la gestion des ports USB : le *hotplug* (cf *hotPnp* sous Windows), le respect de la norme IEEE 1394 (FireWire.i-Link) et le nommage permettant de connecter et de monter automatiquement plusieurs clés USB en même temps (`sysfs` et `udev` permettent d'associer un couple PID/VID à un nom de lecteur spécifiés par des agents et des règles sous `/etc/hotplug`).

Pour le noyau 2.4, la commande `lspci -v|grep -i usb` affiche le type de module chargé :

```
00 :07.2 USB controller : VIA technologies, INC. USB (rev 1a) (prog-if 00 [UHCI])
```

La commande `cat /proc/bus/usb/devices` donne :

```
T: Bus=01 Lev=01 Prnt=01 Port=01 Cnt=01 Dev#= 3 Spd=12 MxCh= 0
D: Ver= 2.00 Cls=00(>ifc ) Sub=00 Prot=00 MxPS=64 #Cfgs= 1
P: Vendor=08ec ProdID=0011 Rev= 2.00
S: Manufacturer=IBM
S: Product=USB 2.0 Memory Key
S: SerialNumber=02169301 0A019FAA
C:* #Ifs= 1 Cfg#= 1 Atr=80 MxPwr= 94mA
I: If#= 0 Alt= 0 #EPs= 2 Cls=08(stor.) Sub=06 Prot=50 Driver=usb-storage
E: Ad=82(I) Atr=02(Bulk) MxPS= 64 Ivl=0ms
E: Ad=01(O) Atr=02(Bulk) MxPS= 64 Ivl=0ms
```

Le résultat indique le VID et le PID du matériel, en déduit qu'il s'agit d'un matériel IBM USB 2.0 Memory Key qui utilise le type de transfert BULK pour dialoguer et transmettre les données.

Mais le périphérique n'est pas encore tout à fait accessible à l'utilisateur. Il est nécessaire pour avoir accès aux données de faire un `mount` par la commande suivante :

```
# mount -v -t vfat /dev/sda1 /mnt/cle
```

qui spécifie le système de fichiers utilisé (VFAT) et le point de montage (`/mnt/cle/`).

Stockage de données

Sous la coque de plastique de la clé se trouve une mémoire flash fonctionnant comme un disque dur amovible, alimentée électriquement par le port USB. C'est une EEPROM à forte densité de mémoire. Les mémoires flash des clés USB ont une capacité de stockage allant de 16 Mo à 2 Go ou plus. Le contrôleur USB de la clé assure le transfert des données entre la mémoire flash et le connecteur USB de l'ordinateur. Il renseigne l'ordinateur sur les caractéristiques de la clé et contient éventuellement un ou plusieurs logiciel(s) particulier(s) appelé(s) *firmware*(s), permettant par exemple de booter sur la clé.

Protection de la clé par mot de passe

L'utilisateur accède à ses données après avoir été identifié par un logiciel (présent sur la clé et sur l'ordinateur). Ce principe impose le déploiement du même logiciel sur tous les ordinateurs du parc informatique, y compris les portables. Par exemple le logiciel FlashKey (chez Freecom) est un exécutable de 224k présent sur la clé et sur le PC – exécutable contenant le mot de passe et exploitable seulement sous Windows.

Partitionnement et protection d'une partie de la mémoire

Le logiciel présent sur la clé USB permet de formater une partie appelée zone de confidentialité accessible par l'utilisateur qui s'identifiera par un mot de passe. Un autre utilisateur de la clé ne pourra accéder qu'à la partie dite libre et ne pourra pas voir la zone de confidentialité (chiffrée par le logiciel propriétaire). Cette partition ne pourra pas être lue sous Linux non plus. Il existe aussi la possibilité de chiffrer et compresser automatiquement tous les fichiers de cette zone. Toujours chez Freecom, le logiciel FlashTools permet de

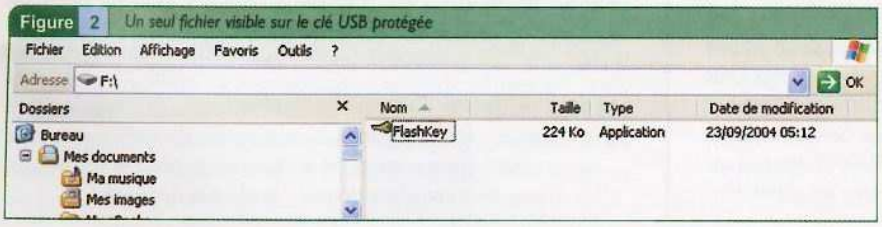


Figure 2 *Un seul fichier visible sur le clé USB protégée*

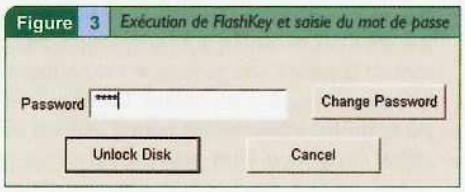
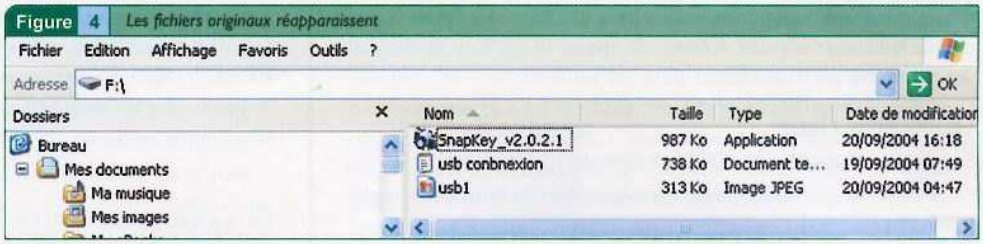


Figure 3 *Exécution de FlashKey et saisie du mot de passe*

formater la clé en FAT classique avec une option permettant cette fonctionnalité de chiffrement/mot de passe.

En mode confidentiel, un seul fichier reste visible de l'explorateur (`flashkey.exe`) (voir figure 2 page précédente), il fait toujours 224k, quelle que soit la taille des fichiers présents sur la clé.

Figure 4 *Les fichiers originaux réapparaissent*



Une fois le mot de passe donné (figure 3), le programme FlashKey présent sur le PC est comparé avec celui stocké sur la clé. Le dialogue est chiffré sur le bus USB. Les fichiers apparaissent alors sur la clé (figure 4). Avec ce genre d'utilitaire, la clé USB reste inexploitable si le PC n'est pas doté du logiciel identique combinant formattage et chiffrement.

Boot sur clé USB

Certaines clés permettent l'amorçage de l'ordinateur. Une condition est toutefois requise : le BIOS de l'hôte doit permettre l'amorçage par les périphériques de stockage USB (cartes mères récentes). Ces partitions de boot donnent accès directement, au démarrage du PC, à un mini système d'exploitation. En fonction de la quantité de mémoire de la clé USB, il permettra d'utiliser des applications plus ou moins évoluées. De cette façon, un utilisateur mobile emploiera son système d'exploitation et ses configurations favorites sur n'importe quel ordinateur. Nous pouvons citer par exemple FLONIX (**www.flonix.com**) qui tient sur une clé de 64 Mo. C'est une distribution gratuite dérivée de Knoppix construite sur Linux.

Identification/Authentification

Utiliser la biométrie (empreintes digitales) est devenu possible sur une clé USB. La clé contient un scanner d'empreintes digitales. Couplés à ce dispositif, un processeur dédié et un système d'exploitation particulier permettent de stocker l'image d'une empreinte digitale et de la comparer à celle de l'utilisateur autorisé. Ce système peut être utilisé pour désactiver l'écran de veille, par exemple.

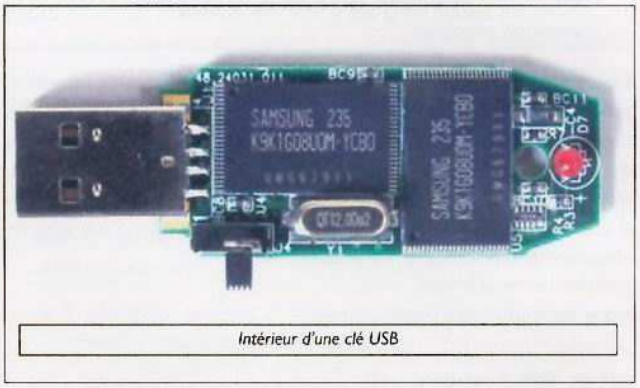
La communication sans fil et le multimédia

Outre les fonctionnalités de lecture des fichiers musicaux MP3, certaines clés se comportent comme de véritables antennes WiFi. Le futur est très prometteur car le marché est présent pour le grand public ayant découvert le monde USB grâce aux modems ADSL, aux appareils photos numériques et autres périphériques stockant des données.

Les risques liés à l'emploi des clés USB

1) Confidentialité et intégrité des données de la clé USB

Les conditions d'environnement ne sont pas gênantes pour utiliser une clé USB, exceptée la condensation. En revanche, les clés USB sont de la taille d'une gomme et l'expérience montre qu'elles sont facilement égarées ou volées. De plus, une fois dérobées, elles peuvent être transportées hors d'un local ou d'une enceinte sans attirer l'attention. Elles peuvent aussi être remises en place sans



Intérieur d'une clé USB

que l'utilisateur ait pu remarquer que les données ont été consultées, copiées ou modifiées. Cet objet personnel a obtenu toute la confiance de son propriétaire. C'est le facteur humain qui rend cette clé vulnérable.

Au niveau mécanique, une technique d'attaque serait de démonter la clé pour accéder aux circuits, récupérer les informations inscrites dessus, puis remettre en place les circuits et refermer l'étui. Cette technique permet de dérober une information protégée sans que son propriétaire soit informé de la compromission. Une équipe de l'université de Reykjavik a tenté cette méthode sur deux types de clés USB : « Aladdin Knowledge System'eToken R1 » et « Rainbow Technologies'ikey 1000, 2000 » (cf. **http://www.atstake.com/research/reports/acrobat/usb_hardware_token.pdf**).

Au niveau électronique, le but est d'accéder au firmware du contrôleur USB. Ce principe permet de connaître précisément les opérations effectuées par le périphérique, les éventuelles erreurs de programmation, fonctions cachées et autres *backdoors* utilisables pour mener une attaque. Cette attaque a été freinée par la technique du security bit : soit il faut démonter le circuit intégré, soit il faut connaître un numéro de 64 bits associé à chaque puce pour pouvoir accéder en lecture ou en écriture au firmware. L'accès à l'EEPROM de la clé Aladdin a été rendu possible avec des soudures et un kit de programmation. Rappelons que l'EEPROM est également sensible à la rémanence électromagnétique et au rayonnement.

Au niveau logiciel, les attaques ont pour objectif l'accès aux données protégées par mot de passe, sans démonter l'étui, uniquement en envoyant des requêtes via un ordinateur hôte. Elles visent à trouver des failles de conception du firmware ou du software utilisés par la clé.

Les deux méthodes d'attaque envisagées sont :

→ d'utiliser un analyseur et un générateur de trafic sur la voie de communication entre la clé et l'ordinateur hôte, et rechercher les éventuelles requêtes donnant accès aux données ;

→ dans le cas de la protection de la mémoire par un mot de passe, forcer ce mot de passe.

Il existe des outils sur le Net permettant de développer et débuguer les périphériques USB sous Windows et sous Linux (cf. références). Par analogie au réseau IP, certains boîtiers ou utilitaires permettent de générer un trafic qui peut contenir à la fois des requêtes écrites selon la norme USB et des requêtes illégales (non prévues par la norme). Le générateur de traffic utilisé sur un hub USB peut usurper une identité PID/VID (analogie au *spoof*), cartographier les périphériques connectés par prise d'empreinte (analogie au *fingerprinting*), voire simuler une cascade de hub USB ou une antenne WiFi (analogie au *pot de miel*). Un *sniffer* USB de type USB Mon ou SNOOPY, installé à l'insu de l'utilisateur, se transforme alors en un spyware USB générant un ficher log de tout le trafic du bus USB.

2) Disponibilité du port USB

En changeant quelques paramètres dans la base de registre Windows, il est possible de verrouiller les ports USB d'un serveur ou d'un PC sensible par exemple. On constate que si sous la clé `HKEY-LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\USBSTOR`, la sous-clé `Start` est modifiée de 3 en 4, l'insertion d'une clé USB ne provoque plus aucun effet.

Une attaque par déni de service consisterait à diffuser sur un réseau informatique un virus ou une installation de logiciel compromis qui procéderait à ce simple remplacement. Plus aucune clé USB ne pourrait être lue dans cette entreprise (cf figure 4).

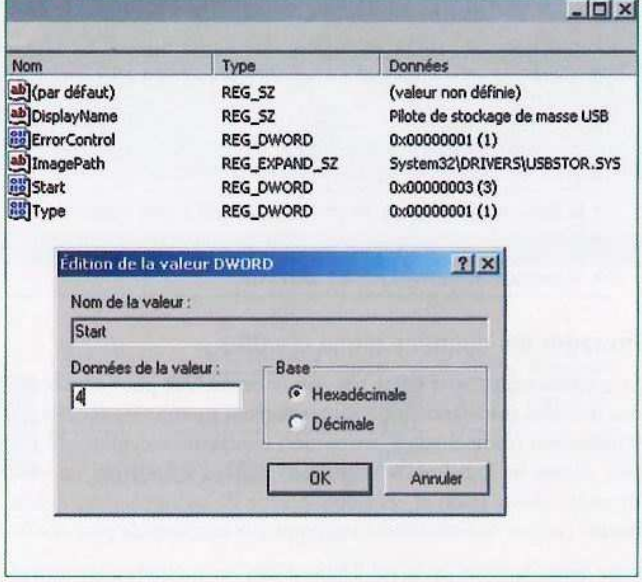
Un autre déni de service consiste à inonder le hub de faux paquets USB simulant la présence de 127 périphériques (PID/VID différents) : plus aucun périphérique ne pourrait alors être reconnu par le système d'exploitation.

3) Corruption du système d'information

Une clé USB se dissimule facilement et permet de transférer en quelques secondes des centaines de Mo de données. Ces caractéristiques permettent d'introduire sur le réseau informatique de l'entreprise des fichiers ou programmes volumineux provenant d'un autre réseau, tels que des jeux, des films, et volontairement ou non, des virus. La clé USB en fait un vecteur de propagation important.

Les ports USB étant maintenant en face avant des ordinateurs, il est facile de booter sur une clé. Pour cela, le BIOS recherche sur les périphériques le MBR sur la tête 0, piste 0 et secteur 1 de la mémoire. Or cette organisation en têtes, pistes et secteurs correspond à un disque dur ou une disquette mais pas à une EEPROM. Pour qu'une clé soit bootable, il faut donc que le firmware du contrôleur USB comprenne un petit logiciel qui simulera un secteur 1, sur la piste 0 d'une tête 0. Le constructeur précise toujours dans les caractéristiques de la clé si elle est bootable ou non. Il ne reste plus qu'à formater la clé en FAT et à copier les fichiers habituels de boot tels que `command.com`, `config.sys` et `autoexec.bat`. Sur un serveur, (Windows 2000 avec NTFS par exemple), il est alors possible de reproduire le cas de la disquette : il suffit de booter sur la clé USB, de lancer le logiciel NTFSPRO pour avoir accès aux données comme par exemple la base des logins et mots de passe, de les copier sur la clé USB et ensuite, d'essayer de découvrir les mots de passe à l'aide d'un cracker.

Figure 4 *Déni de service : modification du registre Windows avec regedit*



L'autorun permet de lancer automatiquement un programme ou l'ouverture d'une fenêtre explorateur dès l'insertion d'un périphérique USB.

Certains fabricants ont mis au point des logiciels (Autorun USB, par exemple) fournis sur le CD d'accompagnement de leurs périphériques.

Le principe de fonctionnement est le suivant :

→ le logiciel est installé sur l'ordinateur client et reste en veille permanente ;

→ un fichier de type `autorun.inf` est présent sur la clé.

À la connexion de la clé, le logiciel scrute le périphérique à la recherche du fichier `.inf` qui renvoie alors sur l'exécution d'un fichier présent sur le périphérique comme un exécutable, un fichier multimédia ou, pourquoi pas, un fichier malicieux.

Par exemple, le texte suivant placé dans `autorun.inf` lance automatiquement un fichier exécutable contenu sur la clé :

```
[AutoRun]
open = superspyware.exe
```

Sécurisation de l'emploi des clés USB

Sensibiliser les administrateurs et les utilisateurs

Pour lutter efficacement contre les risques SSI, il faut avoir conscience des limites et des failles connues des clés USB. C'est pourquoi il est extrêmement important de sensibiliser les administrateurs et les utilisateurs aux risques encourus, à la fois par des démonstrations simples et par la signature d'une reconnaissance de responsabilité des utilisateurs de clés USB en entreprise.

L'engagement doit spécifier les modalités concernant :

→ la perception et la restitution de la clé ;

→ l'usage des clés USB personnelles ;

→ l'introduction de logiciels ou données extérieures via la clé USB ;

→ l'utilisation de la clé professionnelle à l'extérieur de l'entreprise, l'usage personnel ;

→ le respect des règles de conservation de la clé ;

→ le niveau de confidentialité des données pouvant être stockées sur la clé ;

→ les mesures nécessaires afin d'éviter la perte ou le vol (collier, rangement dans un tiroir, etc.) ;

→ le signalement en cas de perte ou de vol à une personne de type OSSI ;

→ le passage systématique à l'antivirus.

Protéger les données d'une clé USB

La solution radicale est de ne pas utiliser de clé USB pour conserver des données confidentielles. Il faut également garantir les conditions d'utilisation (température, protection contre un incendie, etc.). Il faut utiliser les logiciels de protection d'accès à la clé par un mot de passe choisi selon la politique interne de sécurité, c'est-à-dire devant résister suffisamment longtemps aux crackers de passwords.

Pour éviter la perte ou le vol, il faut utiliser les cordons à conserver autour du cou. Pour rendre les données inaccessibles en cas de vol, il faut chiffrer les données stockées. Les logiciels fournis avec la clé sont donc à tester au cas par cas avant l'achat massif de clés USB en entreprise. Le boîtier de la clé peut être enduit d'une colle ou d'un vernis témoignant d'une ouverture physique. Sur le même principe, l'intérieur des clés peut être enduit de résine afin de consolider le contrôleur USB et l'EEPROM. On ne pourra pas les retirer sans les casser.

Côté rémanence magnétique, le phénomène est amplifié lorsqu'une donnée est stockée longtemps sur le même emplacement mémoire. Plus la mémoire est dense, plus il sera difficile d'y retrouver une information effacée.

Se protéger des paquets USB illégaux reste impossible car seul le constructeur peut modifier le firmware et rien ne nous garantit que le constructeur ne s'est pas laissé le droit d'utiliser des requêtes lui permettant de contourner un mot de passe.

Assurer la disponibilité du port USB

Le blocage du port USB est possible au niveau du BIOS et du système d'exploitation. Le BIOS doit être protégé au minimum par un mot de passe et la configuration du système d'exploitation doit être sauvegardée (prise d'empreinte de type TRIPWIRE de la base des registres Windows par exemple). Inutile de rappeler qu'il faut un BIOS, un antivirus et un système d'exploitation à jour.

Protéger son système d'information

Sur un serveur, il est judicieux de condamner physiquement le port USB, de le désactiver dans le BIOS et de ne pas démarrer le module correspondant du système d'exploitation. Pour lutter contre l'introduction ou le vol de données via une clé USB, il existe des solutions logicielles.

De nombreux éditeurs ou sharewares proposent de bloquer les ports USB au niveau du système d'exploitation, comme Folder Security Personnal 2.50 et Device Lock. L'administrateur d'une machine ou d'un domaine peut interdire ou limiter les connexions des clés USB ou autres périphériques de stockage à distance, selon des plages horaires. Le logiciel Reflex Disknet Pro 4.40 va jusqu'à interdire l'exécution des fichiers `autorun.inf`, permet de scanner les clés USB par un antivirus dès leur connexion et journalise les connexions et déconnexions des clés USB. Le coût de déploiement n'est pas négligeable sur un parc informatique important.

Sous Linux, le module PAM `pam-usb` transforme une clé USB en une véritable clé pour s'authentifier sur sa machine. Son fonctionnement est simple : à l'aide de l'utilitaire `usbadm`, on génère et stocke un couple de clés DSA sur sa clé USB et sur son ordinateur. L'accès à la machine est accepté ou refusé selon le résultat du challenge entre la clé publique et la clé privée. Le principe est utilisable par le programme `login`. Il permet de chiffrer la clé privée avec un algorithme symétrique et supporte des *access lists* de numéros de série pour empêcher l'accès aux clés inconnues : une sorte de filtre USB.

CONCLUSION

Il n'existe à ce jour pas de prise de conscience du risque concernant l'utilisation des clés USB en milieu professionnel. Les spécifications de la norme USB ne sont pas tenues secrètes. La simple connaissance des mécanismes employés par les clés USB est très importante. En effet, elle met en lumière les failles que pourraient utiliser des personnes malveillantes.

Les administrateurs systèmes doivent comparer les dangers encourus avec leurs objectifs de sécurité informatique et en tirer des conséquences, c'est-à-dire employer les méthodes de sécurisation proposées lorsqu'ils le jugent nécessaire.

Les éditeurs de logiciels se réveillent et proposent maintenant des utilitaires mettant en œuvre une réelle politique de sécurité des clés USB. Votre clé USB n'est vraiment pas un jouet.

Bibliographie

- **http://www.lvr.com/usbhtm/** : sniffer USB, scripts utiles en Perl, générateur de trafic USB.
- **http://www.linux-usb.org** : Linux USB Project
- **http://www.linux.org** : HOWTO USB Flash Memory
- **http://linux-hotplug.sourceforge.net** : hotPnP sour Linux noyau 2.6
- **http://www.usb.org** : norme USB
- Etude E.S.A.T. : *La clé USB* - N. Malbec et S. Marziou
- MSDN : *Plug and Play for Windows 2000, Identifiers for USB devices*
- **http://www.hhdsoftware.com/usbmon.html/** : USB Monitor 2.37 sous Windows
- **http://sourceforge.net/projects/usbsnoop/** : SNOOPY PRO 0.22 sous Windows
- Folder Security Personnal 2.50, Device Lock et Reflex Disknet Pro 4.40
- **http://www.sysinternals.com** : NTFSPRO

Des erreurs dans mon code sécurisé ? ah bon ?! où ça ?

Axelle Apvrille

Junior est un jeune programmeur dévoué de la société Pigeons S.A. Sans être un « expert » sécurité – c'est loin d'être son métier – il aime se tenir au courant des dernières nouveautés et failles en matière de sécurité. Bref, sans connaître à fond les détails, Junior n'est pas plus bête qu'un autre – voire moins – et a déjà lu quelques articles de base sur les débordements de buffer ou la cryptologie. Ces connaissances lui ont valu la casquette de « Monsieur Sécurité », et c'est donc pour cela que, ce matin, son chef est venu lui demander d'implémenter un mécanisme d'authentification simple, mais sûr.

Figure 1 *Protocole d'authentification par défi*

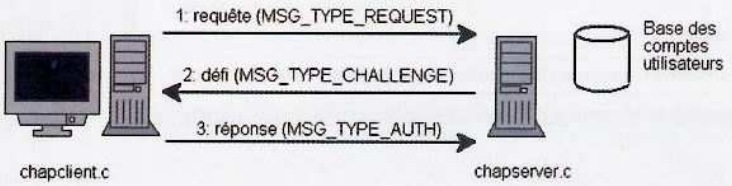
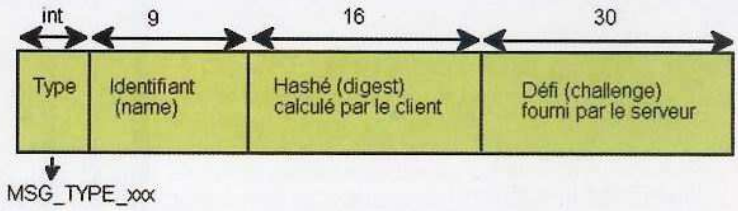


Figure 2 *Format des messages d'authentification (struct message)*



Aussitôt, Junior s'est plongé dans les quelques notes qu'il prend parfois la peine de gribouiller lorsqu'il tombe sur quelque chose d'intéressant sur le Web ou dans un bon livre. Finalement, il opte pour le très répandu système d'authentification par mot de passe avec défi illustré à la Figure 1. C'est simple, mais au moins le mot de passe ne transite pas en clair, et le procédé résiste aux rejeux.

Pour s'authentifier, l'utilisateur (client) envoie une requête d'authentification au serveur (étape 1). Le serveur tire alors aléatoirement un nouveau « défi » et l'envoie au client (étape 2). Ce dernier prouve son identité en calculant une réponse à partir du défi (qui change à chaque demande) et de son mot de passe. Cette preuve consiste en un condensé qui est le résultat d'une fonction de *hashage*. Histoire de ne pas réinventer la roue, Junior base son implémentation sur la librairie crypto d'OpenSSL (fonctions `EVP_xxx`) [1]. Le résultat est renvoyé au serveur (étape 3). Enfin, le serveur vérifie la réponse qui lui est fournie. Pour cela, il récupère dans une base de données le mot de passe de l'utilisateur, calcule la réponse attendue et la compare à celle de l'utilisateur. Si cela concorde, l'authentification se termine avec succès.

Junior définit ensuite le format des messages échangés. Pour se simplifier la tâche, il utilise un seul format de paquet, de taille **fixe** pour les 3 messages (voir Figure 2). La transmission des paquets entre client et serveur repose sur une interface qui exporte les fonctions `sendMsg(message à envoyer, destinataire)` – pour envoyer un message au destinataire donné, et `recvMsg(destinataire)` pour récupérer un message envoyé à l'intention du destinataire indiqué. Un *destinataire* est soit le client (`RECIPIENT_CLIENT`) soit le serveur (`RECIPIENT_SERVER`).

L'implémentation de cette API est hors sujet dans cet article. On peut imaginer qu'elle repose sur des *sockets*, *named pipes* ou par-dessus un autre protocole spécifique.

Les messages (`struct message` dans le code) sont donc constitués de :

- → un type (`MSG_TYPE_xxx`), codé sur 4 octets.
- → le nom de l'utilisateur (appelé `identifiant`), sur 9 octets. Ce nom doit être terminé par le caractère `'\0'`, on dispose donc en fait de 8 caractères utiles.
- → un hashé, calculé par le client, sur 16 octets. Ce hashé n'est présent que dans les messages de type `MSG_TYPE_AUTH`, et est ignoré dans les autres cas.
- → un défi, généré aléatoirement par le serveur, de 30 octets. Ce défi n'est présent que dans les messages de type `MSG_TYPE_CHALLENGE`. On ignore le champ dans les autres cas.

Plus précisément, la création du message de défi du serveur (`MSG_TYPE_CHALLENGE`) et la création de la réponse du client (`MSG_TYPE_AUTH`) sont détaillées respectivement dans le code n°1 et n°2 page suivante.

❑ code n°1

```
1   static void buildMsgChallenge(struct message *msg){
      char *challenge;
      time_t current;
5     msg->type = MSG_TYPE_CHALLENGE;
      time(&current);
      challenge = ctime(&current);
      strncpy(msg->challenge,challenge,LEN_CHALLENGE);
    }
```

Création du message de défi du serveur (extrait de chapserver.c)

❑ code n°2

```
1   static int buildMsgAuth(struct message *msg, char *name, char *passwd,
char *challenge){
      char *todigest;
      int len_passwd, len_challenge, len_digest;
      EVP_MD_CTX mdctx;
5
      todigest = NULL;
      msg->type = MSG_TYPE_AUTH;
      strncpy(msg->name,name,LEN_NAME);

10    len_passwd = strlen(passwd);
      len_challenge = strlen(challenge);
      todigest = (char*) malloc(len_passwd+len_challenge);

      memcpy(todigest,passwd,len_passwd);
15    memcpy(todigest+len_passwd,challenge,len_challenge);

      EVP_DigestInit(&mdctx,EVP_md5());
      EVP_DigestUpdate(&mdctx,todigest,len_passwd+len_challenge);
      EVP_DigestFinal(&mdctx,&msg->digest[0],&len_digest);
20
      free(todigest);  return 0;
    }
```

Création de la réponse du client (extrait de chapclient.c)

La fonction `buildMsgChallenge` reçoit en entrée un message alloué, mais vide. Les champs `name` (nom d'utilisateur), `digest` (hashé) et `challenge` (défi) sont remplis de 0x00.

La fonction `buildMsgAuth` reçoit également en entrée un message alloué mais vide (même conditions que pour `buildMsgChallenge`). De surcroît, elle reçoit en entrée :

→ le nom que l'utilisateur a entré au clavier. Ce nom fait au maximum 9 octets (éventuellement moins), et on est sûr qu'il se termine par un caractère '\0'.

→ idem pour le mot de passe : au maximum 9 octets, et terminé par le caractère '\0'.

→ un défi : celui-ci est lu du message envoyé par le serveur (`MSG_TYPE_CHALLENGE`). La façon de constituer ce message est détaillée dans le code n°1.

Pour la forme, la fonction retourne la valeur 0 (quoi qu'il arrive). Le code de retour n'est pas utilisé pour l'instant.

Du côté du serveur, Junior modélise la base de données des comptes utilisateurs au moyen d'une variable globale `gDb` de type *struct database* au format suivant :

→ le nom de l'utilisateur (8+1=9 octets).

→ le mot de passe de l'utilisateur (8+1=9 octets). Comme pour le nom, on considère qu'un mot de passe doit être terminé par '\0', on dispose donc en fait de 8 caractères utiles pour le mot de passe.

→ le défi (30 octets).

→ un état, codé sur 4 octets, qui prend la valeur `CHALLENGE_NEW` (égale à 1) lorsqu'une authentification est en cours (un défi a été généré, mais le serveur n'a pas encore reçu de réponse correspondante) et `CHALLENGE_OLD` (égale à 0) sinon.

La recherche de la ligne de tableau correspondant à un utilisateur donné est alors très simple. Il suffit de parcourir tout le tableau `gDb` et de s'arrêter dès qu'on trouve le nom recherché. Ceci est détaillé dans la fonction `lookupName` du code n°3.

Cette fonction reçoit en entrée le nom de l'utilisateur à rechercher. Ce nom fait au maximum 9 octets et est forcément terminé par un '\0'.

❑ code n°3

```
1  static int lookupName(char *name){
     int i;
     for (i=0; i< NB_USERS; i++){
       if (strncmp(gDb[i].name,name,LEN_NAME) == 0) {
5        return i;
       }
     }
     printf(«Utilisateur introuvable.\n»);
     return -1;
10 }
```

Recherche d'un utilisateur donné dans la base des utilisateurs (extrait de chapserver.c)

Enfin, on fournit ci-après (code n°4) le code de validation utilisé par le serveur. Il s'agit de la fonction `doCheck`, qui prend en entrée le message `MSG_TYPE_AUTH` renvoyé par le client. La fonction retourne 1 si l'authentification est validée, -1 si une erreur survient et qu'elle est refusée.

❑ code n°4

```
1  static int doCheck(struct message *authmsg){
     int len_passwd, len_challenge, len_digest, index, result;
     char *todigest;
     unsigned char digest[LEN_DIGEST];
5    EVP_MD_CTX mdctx;
     todigest = NULL;
     result = -1;
     index = lookupName(authmsg->name);
10   if (index == -1) return -1;
     if (gDb[index].status != CHALLENGE_NEW) {
       return -1;
     }

15   len_passwd = strlen(gDb[index].passwd);
     len_challenge = strlen(gDb[index].challenge);
     todigest = (char*) malloc(len_passwd+len_challenge);

     memcpy(todigest,gDb[index].passwd,len_passwd);
20   memcpy(todigest+len_passwd,gDb[index].challenge,len_challenge);
     EVP_DigestInit(&mdctx,EVP_md5());
     EVP_DigestUpdate(&mdctx,todigest,len_passwd+len_challenge);
     EVP_DigestFinal(&mdctx,digest,&len_digest);
25
     if (memcmp(authmsg->digest, digest, len_digest)==0){
       printf("Authentification: SUCCES.\n");
       result = 1;
     } else {
30     printf("Authentification: REFUSE.\n");
     }

     gDb[index].status = CHALLENGE_OLD;
     free(todigest);
35   return result;
   }
```

Validation du message de réponse du client (extrait de chapserver.c)

ÉNONCÉ

L'énigme de ce numéro consiste à trouver les erreurs de sécurité dans l'implémentation de Junior.

Il peut s'agir :

→ d'erreur de conception : l'approche choisie contient en elle-même des failles de sécurité ;

→ d'erreur d'implémentation : les principes énoncés lors de la conception sont mal adaptés dans le code. Cela peut provenir d'une ambiguïté de la conception, de mauvaise compréhension, ou d'une erreur plus bête ;

→ d'erreur de programmation : le code comporte des bugs (gestion erronée de cas d'erreurs, oubli de libérer l'espace mémoire…) qui donnent lieu à des failles de sécurité.

En revanche,

→ On ne considérera que les attaques « logiques », et non pas les attaques basées sur le hardware.

→ On ne prêtera attention qu'aux erreurs présentes dans les portions de code fournies ci-dessus. Si néanmoins vous souhaitez compiler les fichiers sources complets de Junior, vous pouvez les télécharger sur le site de MISC [2].

→ On ignorera les erreurs qui n'ont pas d'impact sur la sécurité. Attention cependant car les erreurs les plus anodines cachent parfois de grandes failles…

Je remercie tout particulièrement Philippe Biondi, Christophe Grenier, ainsi que Frédéric Raynal pour leurs suggestions et remarques.

Vous trouverez la solution à cette enigme dans le prochain No de Misc, à paraître début janvier 2005

Références

[1] OpenSSL, **http://www.openssl.org**

[2] Sources complètes : **http://www.miscmag.com/articles/16-MISC/Enigme/**

[3] David Wheeler, *Secure Programming for Linux and Unix HOWTO*, **http://www.dwheeler.com/secure-programs/**

[4] Greg Hoglund, Gary McGraw, *Exploiting Software: How to break code*, Ed. Addison Wesley, 2004, ISBN 0-201-78695-8

[5] Michael Howard, David Leblanc, *Writing Secure Code*, Microsoft Press, 2002, ISBN 0-772-86435-4

Analyses de codes malveillants pour mobiles : le ver CABIR et le virus DUTS

En juin et juillet 2004, deux nouvelles infections informatiques ont fait leur apparition et ont été présentées comme des innovations. Si les plate-formes visées sont effectivement inhabituelles — certains téléphones portables pour le ver CABIR et des ordinateurs de poche pour le virus DUTS — en réalité, il ne s'agit que de codes viraux tout à fait communs, développés pour des environnements en apparence « exotiques ». Dans la mesure où ces environnements sont comparables à n'importe quel ordinateur traditionnel, l'apparition de codes viraux spécifiques est somme toute logique. Dans cet article, nous présentons le ver CABIR et le virus DUTS et montrons qu'ils ne diffèrent fondamentalement pas des autres vers ou virus connus, même si le résultat constitue toutefois, dans les deux cas, une prouesse sur le plan de l'implémentation.

Le ver CABIR

Présentation générale

Ce ver, qui existe en deux versions, a été conçu par un des membres du groupe 29A [1] et envoyé en juin 2004 à la société Kaspersky Labs. Présenté, non sans un certain battage médiatique, comme un nouveau concept de virus (*proof-of-concept*), le ver CABIR n'est en fait qu'un banal ver d'emails (voir [2] pour une définition des différents types de vers), tout à fait commun. La seule différence vient du fait de son portage sur une plate-forme inhabituelle : les téléphones portables. Pendant longtemps, la notion de virus pour téléphones cellulaires a excité les esprits et sa faisabilité sujette à caution. En réalité, CABIR n'est que la concrétisation d'une idée logique dès lors que les téléphones mobiles possèdent un environnement informatique analogue — même s'il est encore un peu frustre — à celui de n'importe quel ordinateur.

Ce ver a pour cible les téléphones portables fonctionnant avec le système d'exploitation Symbian [3]. Actuellement, près d'une trentaine de modèles en sont équipés et ce nombre devrait sans nul doute s'allonger avec le succès de ce système d'exploitation. Il faut noter que tout système mobile doté du système Symbian est potentiellement vulnérable (smartphones ou autres appareils communiquant sans fil).

Ce système d'exploitation [3] est un système orienté environnements mobiles et sans fil. Ses caractéristiques sont assez nombreuses et lui ont assuré une position largement prédominante dans ce secteur. Doté d'un environnement graphique ergonomique et proche, dans l'esprit, de celui des systèmes d'exploitation que nous utilisons tous sur nos ordinateurs traditionnels, il a été très facilement adopté par les utilisateurs.

Ses principales caractéristiques sont les suivantes :

- → supporte le développement d'applications en Java, C++ et gère différents formats de données ;
- → intègre un noyau multitâche ainsi que de nombreux services et applications (graphiques, gestion de données, etc.) ;
- → conforme et interopérable avec de nombreux standards : IPv4, IPv6, Bluetooth, Java, WAP, SyncML, etc.
- → gestion modulaire par API (*Application Programming Interfaces*) communes à tous les environnements Symbian.

CABIR se présente sous la forme d'un fichier au format SIS appelé `caribe.sis`, d'une taille de 15092 octets (la seconde version connue fait 15104 octets) contenant :

- → l'exécutable principal `caribe.app` (respectivement 11932/11944 octets) ;
- → des données d'identification système contenues dans le fichier `flo.mdl` (2544 octets) ;
- → un fichier ressources `caribe.rsc` (44 octets).

Une fonction `autostart` intégrée permet d'exécuter le fichier `caribe.app` dès que le fichier `caribe.sis` est installé. Enfin, le ver utilise les fonctionnalités de communication offertes par la norme Bluetooth pour se propager à chaque connexion. Les cibles sont les appareils compatibles avec cette norme, laquelle doit être de plus activée.

La propagation du ver

Considérons un tel appareil déjà infecté. Le ver devient actif à chaque mise sous tension. Il scanne toutes les connexions Bluetooth actives. Il choisit la première rencontrée et tente d'envoyer le fichier `caribe.sis` à l'appareil connecté (encore une fois, il ne s'agit pas forcément d'un téléphone mobile).

Ce dernier reçoit alors une notification de réception de message comme illustré dans la figure 1.

Une fois que le destinataire a confirmé la réception du message, il lui est demandé s'il souhaite installer le fichier `caribe.sis` (figure 2).

Si l'utilisateur choisit d'installer ce fichier, un message d'avertissement est affiché, lui indiquant d'éventuels problèmes de sécurité (figure 3).

En cas de confirmation, le ver s'installe. Le mode de propagation du ver CABIR est donc similaire à celui d'un ver d'emails présent dans la pièce jointe. Le ver ne se propage que si l'utilisateur inconscient active le fichier envoyé. Seul l'environnement change : CABIR n'est pas un ver innovant.

Le mécanisme d'infection

Une fois activé, le ver installe ses fichiers dans différents répertoires systèmes, de la manière suivante :

```
c:\system\apps\caribe\caribe.rsc
c:\system\apps\caribe\caribe.app
c:\system\apps\caribe\flo.mdl
c:\system\symbiansecuredata\caribesecuritymanager\caribe.app
c:\system\symbiansecuredata\caribesecuritymanager\caribe.rsc
c:\system\recogs\flo.mdl
```

Le répertoire `SYMBIANSECUREDATA` créé par le ver est caché (invisible par l'utilisateur). De plus, en cas de suppression des fichiers viraux

Eric Filiol - **efiliol@esat.terre.defense.gouv.fr**
Ecole Supérieure et d'Application des Transmissions
Laboratoire de cryptologie et de virologie
http://www-rocq.inria.fr/codes/Eric.Filiol/index.html

Figure 1
Envoi d'un message infecté par CABIR

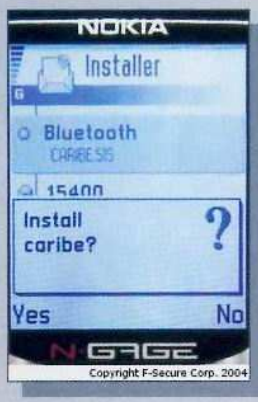


Figure 2
Prompt d'installation de CABIR

Figure 3
Message d'avertissement de sécurité

Figure 4
Message d'installation de CABIR

du répertoire `c:\system\apps`, le ver reste actif dans le système (phénomène de résidence/persistance).

Lorsqu'il est lancé, le ver affiche le message suivant contenant son nom, l'alias de l'auteur (VZ) et le groupe auquel ce dernier appartient (29A) (figure 4).

Enfin, le ver reconstruit un nouveau fichier infecté `caribe.sys` et tente de propager l'infection vers un nouvel appareil.

Détection et désinfection

La détection de CABIR est assez aisée, même manuellement. La plupart des antivirus diffusant une version pour mobiles ont désormais intégré le ver dans leur base de signatures.

La désinfection est également facile :

→ soit automatiquement, avec un utilitaire fourni par un éditeur d'antivirus,

→ soit manuellement, en effaçant les fichiers mentionnés plus haut (un gestionnaire de fichiers doit cependant être présent). La suppression du fichier `caribe.rsc` est impossible tant que le ver est actif. Il faut donc effacer préalablement les autres fichiers viraux, dans tous les répertoires où ils se trouvent, redémarrer ensuite l'appareil et finalement effacer le fichier `caribe.rsc`.

Il est probable que d'autres codes malveillants mettront en œuvre des techniques destinées à rendre plus difficiles ces opérations de détection et désinfection comme c'est le cas pour les autres virus et vers.

Le virus DUTS

Cette analyse s'appuie sur l'analyse directe du code source.

Ce virus vise, comme le ver CABIR, un nouveau type de plate-forme : les ordinateurs de poche (pocketPC) utilisant le processeur ARM [4]. En réalité, ces environnements ne se distinguent pas fondamentalement des ordinateurs traditionnels. Ils disposent, en particulier, d'une version embarquée du système d'exploitation Windows : WinCE. L'apparition de virus n'est donc pas un événement exceptionnel. À noter que les téléphones portables supportant l'environnement PocketPC à base d'ARM sont également vulnérables face à ce virus.

Signalons que le véritable nom de ce virus, `Wince_dust` ou DUST tout simplement, est un hommage au roman de science-fiction *Permutation City* écrit par Greg Egan. Le code contient la référence suivante (non affichée mais visible lors de l'édition avec un éditeur hexadécimal) :

```
; a little reminiscence of my beloved book - Greg Egan's Permutation City
DCB     «This code arose from the dust of Permutation City»
ALIGN   4
```

Les gens de Fsecure ont bizarrement choisi de l'appeler DUTS. Un autre message caché figure dans le code :

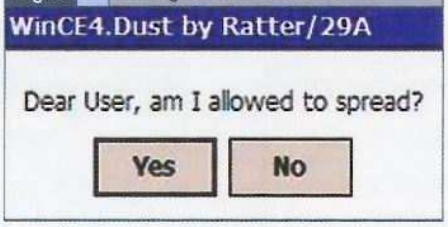
```
; Just a little greeting to AV firms :-)
DCB     «This is proof of concept code. Also, i wanted to make avers happy.»
DCB     «The situation when Pocket PC antiviruses detect only EICAR file had»
DCB     « to end ...»
ALIGN   4
```

DUTS a été, comme CABIR, développé par le groupe 29A comme preuve de faisabilité et communiqué à la société d'antivirus Fsecure à la mi-juillet 2004. Notons au passage que cette contribution d'un groupe connu de développeurs de virus, très utile, a permis de démontrer en pratique un risque attendu. Cela illustre le fait qu'une partie de cette communauté — les programmeurs de virus — a fait preuve d'une attitude responsable et a un rôle essentiel, constructif à jouer.

DUTS est un programme de 1520 octets de long, écrit en assembleur ARM. Il s'agit d'un virus agissant par ajout de code, en fin du fichier cible. Le mode de propagation est donc simple et se fait par échange de fichier (quel que soit le mode).

Le virus est activé lorsqu'un fichier infecté est exécuté. À ce moment là, le virus affiche un message demandant l'autorisation de s'installer (voir figure 5). Il s'agit de la volonté de son auteur, mais un virus plus méchant n'aurait pas eu cette « politesse ».

Figure 5 *Message d'installation de DUTS*



Le code de la fonction concernée (`ask_user`) est le suivant :

```
ask_user        PROC
        str     lr, [sp, #-4]!

        mov     r0, #0
        adr     r1, text
        adr     r2, caption
        mov     r3, #4

        mov     lr, pc
        ldr     pc, [r11, #-12]

        cmp     r0, #7

        ldr     pc, [sp], #4
        ENDP
```

Les chaînes de caractères `text` et `caption` sont définies ensuite (format unicode) :

```
        ; WinCE4.Dust by Ratter/29A
caption DCB     «W», 0x0, «i», 0x0, «n», 0x0, «C», 0x0, «E», 0x0, «4», 0x0
        DCB     «.», 0x0, «D», 0x0, «u», 0x0, «s», 0x0, «t», 0x0, « «, 0x0
        DCB     «b», 0x0, «y», 0x0, « «, 0x0, «R», 0x0, «a», 0x0, «t», 0x0
        DCB     «t», 0x0, «e», 0x0, «r», 0x0, «/», 0x0, «2», 0x0, «9», 0x0
        DCB     «A», 0x0, 0x0, 0x0

        ALIGN   4

        ; Dear User, am I allowed to spread?

text    DCB     «D», 0x0, «e», 0x0, «a», 0x0, «r», 0x0, « «, 0x0, «U», 0x0
        DCB     «s», 0x0, «e», 0x0, «r», 0x0, «,», 0x0, « «, 0x0, «a», 0x0
        DCB     «m», 0x0, « «, 0x0, «I», 0x0, « «, 0x0, «a», 0x0, «l», 0x0
        DCB     «l», 0x0, «o», 0x0, «w», 0x0, «e», 0x0, «d», 0x0, « «, 0x0
        DCB     «t», 0x0, «o», 0x0, « «, 0x0, «s», 0x0, «p», 0x0, «r», 0x0
        DCB     «e», 0x0, «a», 0x0, «d», 0x0, «?», 0x0, 0x0, 0x0
        ALIGN   4
```

En cas d'accord de l'utilisateur, le virus procède alors à l'infection de la manière suivante :

→ le virus cherche à infecter tous les fichiers de type `*.exe` (fonctions `FindFirstFile/FindNextFile` et procédure `find_files_iterate`) présents dans le répertoire courant (en réalité, il y a un bogue à ce niveau ; en effet, la notion de répertoire courant n'existe pas avec WinCE et il est nécessaire d'utiliser des chemins absolus pour désigner les fichiers. Seuls les fichiers du répertoire racine seront infectés).

→ la routine de recherche vérifie que le fichier n'est pas d'une taille supérieure à 4096 octets. Si cela est le cas, la fonction `infect_file` est appelée pour procéder à l'infection proprement dite.

→ dans la procédure `infect_file`, est tout d'abord vérifié que le fichier n'a pas déjà été infecté (contrôle de la surinfection). Pour cela, lors de l'infection, le virus installe sa propre signature (ou marqueur d'infection) (chaîne de caractères << `atar` >> dans le champ de l'en-tête d'exécutable de la version de Windows) :

```
ldr     r0, [r2, #0x4c]  ; chargement du champ de l'entête.
ldr     r1, =0x72617461  ; chargement de la chaine «atar»
cmp     r0, r1           ; comparaison
......
ldr     r0, =0x72617461
str     r0, [r4, #0x4c]  ; stockage de la signature lors de l'infection
```

Le but de ces deux vérifications (taille et signature) est d'obtenir un minimum de furtivité, notamment par limitation de taille. Si aucun contrôle de la surinfection n'était réalisé, la taille du fichier infecté augmenterait à chaque nouvelle infection.

→ le code viral est ajouté à la fin et la dernière section est rendue accessible en lecture/exécution afin d'étendre l'action du point d'entrée viral installé, lui, au début du code (modification et actualisation de l'en-tête PE).

Au final, ce virus présente un risque quasi-nul dans sa forme présente. Gageons que des variantes, plus agressives, feront leur apparition dans peu de temps. Les antivirus existant pour ces environnements ont été naturellement mis à jour.

Conclusion

Ces deux infections informatiques, présentées comme des *proof-of-concept* ne sont en fait que des portages de techniques virales connues sur de nouvelles plates-formes. En termes d'algorithmique virale, ils n'offrent aucun caractère innovant, contrairement à ce qu'a pu laisser penser un certain battage médiatique — qui n'est d'ailleurs pas le fait des auteurs de ces codes. La performance des auteurs vient plutôt sur portage de techniques connues vers des environnements exotiques.

En revanche, CABIR et DUTS nous enseignent que les réflexes de l'utilisateur en termes de sécurité informatique doivent être étendus à ces environnements en apparence nouveaux. Cela réclame la même vigilance face à des messages suspects, des mails avec pièces jointes ou des programmes d'origine douteuse ou inconnue. Au-delà de leur aspect anodin ou ludique, ces environnements sont comparables à nos ordinateurs. Le prochain danger viendra de virus qui, profitant des possibilités de connexion avec des ordinateurs traditionnels, s'y propageront par l'intermédiaire des téléphones et autres environnements mobiles.

Références

[1] **http://29Alabs.host.sk**

[2] E. Filiol - *Les virus informatiques : théorie, pratique et applications* - Collection IRIS, Springer, 2003.

[3] **http://www.symbian.com**

[4] **http://www.arm.com**

Eléments de sécurisation des PABX

Abstract

L'analyse de sécurité des PABX a déjà été abordée dans MISC sous l'angle du test d'intrusion, qu'il soit interne ou externe [1]. Cet article a pour objectif de présenter des éléments permettant d'effectuer une analyse de la configuration d'un PABX et aussi de le sécuriser.

Résumé de l'épisode précédent

Les familles de menaces et vulnérabilités génériques ne seront pas rappelées dans le détail car elles ont déjà été décrites dans le MISC ou dans le document du CLUSIF [2]. Il en sera de même au sujet des différentes techniques d'intrusion (test de phoning, de messagerie vocale, etc.).

De manière très concise et donc imparfaite, un autocommutateur téléphonique constitue potentiellement une cible présentant, au moins, deux intérêts :

→ ses ressources « Télécom » qu'un attaquant chercherait à détruire, entraver ou détourner pour, par exemple, « alléger » sa facture personnelle, mais aussi pour bien d'autres motifs ;

→ son éventuelle interconnexion à un réseau informatique qui le rendrait intéressant en vue d'effectuer un rebond vers ce réseau.

Cet article a donc pour objectif de donner des éléments permettant d'analyser la configuration d'un autocommutateur. Cependant, l'offre sur ce type de matériel étant assez diversifiée et leurs fonctionnalités très nombreuses, il est impossible d'être exhaustif et les noms des fonctions pourront varier d'un matériel à l'autre. Cependant, l'article tâchera de reprendre la terminologie des paramètres rencontrés dans les interfaces courantes d'administration.

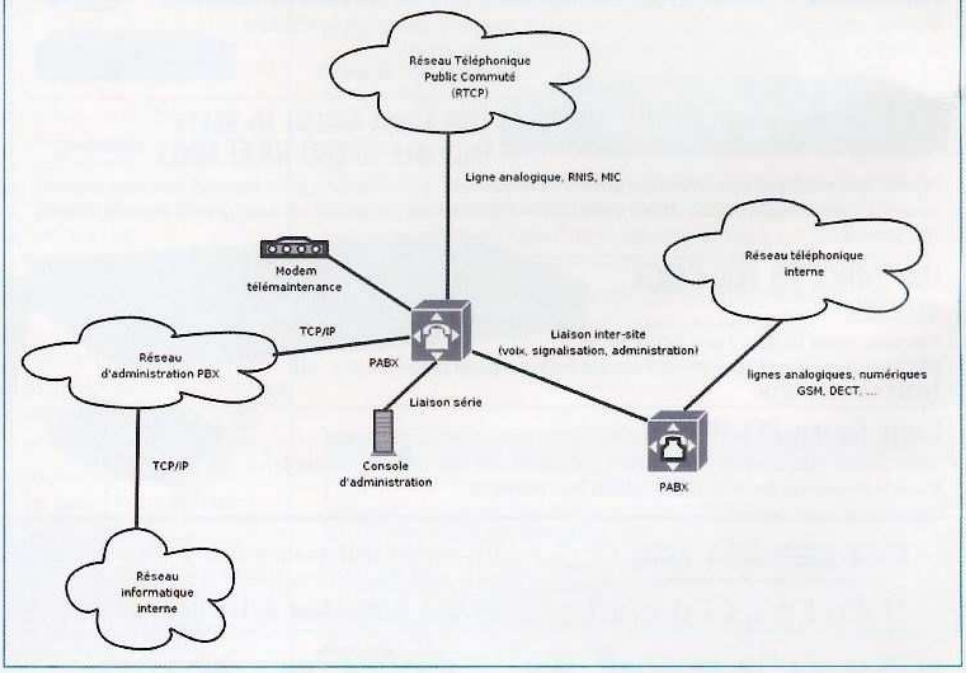
La problématique VoIP étant abordée dans un autre article de ce numéro, il sera présenté un PABX « old fashion » (MIC bien classique) car la durée de vie de ces équipements est assez importante et ils sont encore souvent présents. De plus, la VoIP, qu'elle soit du côté opérateur ou du côté des postes internes, n'est qu'un nouveau protocole de communication, avec ses propres besoins de sécurité venant s'ajouter aux méthodes de sécurisation classiques et non les remettre complètement en question.

Enfin, bien qu'on parle désormais de la convergence « voix/données », n'oubliez jamais la dualité d'un PABX : à la fois système informatique, mais aussi et surtout système Télécom. Ces équipements ont donc de fortes spécificités propres aux « téléphonistes » qu'un « informaticien » ne pourra totalement maîtriser (et réciproquement), ce qui rend les deux spécialités fortement complémentaires. Pour toute manipulation sur un PABX, il est donc préférable de toujours le faire avec l'assistance d'un « téléphoniste » : soyez curieux, mais évitez les expérimentations sur des systèmes en production...

Les canaux d'accès à un PABX

Comme vous pouvez le voir sur la figure 1, loin d'être exhaustive en termes de possibilités d'interconnexion, un PABX est très communicant et utilise un grand nombre de protocoles : MIC, RNIS, interfaces analogiques, TCP/IP, X25, VoIP, LIA, etc. Il n'est pas rare que le maillage d'un réseau téléphonique soit très complexe et possède une étendue internationale dans certaines grosses structures. Ceci constitue des cibles de choix pour les attaquants cherchant à rester les plus discrets possible, bénéficier d'un réseau de portée mondiale, rebondir sur d'autres réseaux en bénéficiant de l'écran que constitue le réseau de départ.

Figure 1 *Réseau de télécommunication*



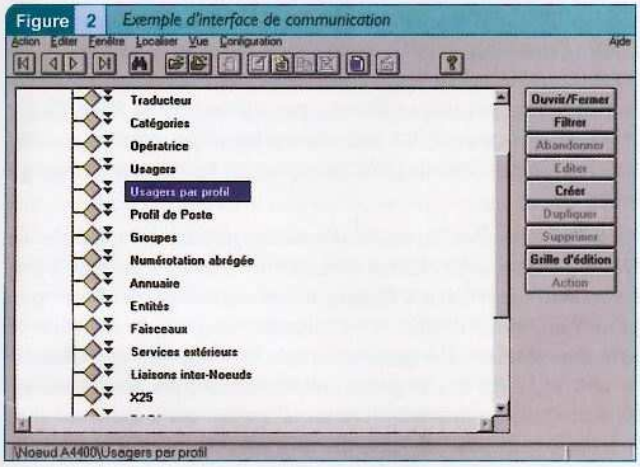
Laurent Estieux
laurent.estieux@free.fr

Chacun de ces protocoles comporte ses vulnérabilités intrinsèques, mais les pirates se concentrent sur des canaux particulièrement intéressants.

La console d'administration standard

Cette console est une interface de paramétrage du PABX constituée d'une application graphique (cf. figure 2) installée sur un PC sous un OS standard, qu'il faut sécuriser, et communiquant avec le PABX bien souvent par une liaison série (ou Ethernet).

Figure 2 *Exemple d'interface de communication*



Ces interfaces manipulent généralement une structure arborescente des paramètres de configuration du PABX rappelant une base de registre. L'ensemble de ces paramètres peut aussi être manipulé localement sur le PABX grâce à des outils spécifiques au PABX.

La ligne de télémaintenance

La société de maintenance du PABX possède souvent une ligne dédiée à la maintenance du PABX pour effectuer des opérations d'administration distantes. Le modem de télémaintenance est pratiquement toujours allumé, très souvent directement géré par le PABX (donc débouchant sur un « pseudo shell » d'administration) et les authentifiants triviaux. La découverte du numéro de cette ligne de télémaintenance amène souvent rapidement à un accès privilégié à l'autocommutateur.

Pour éviter cela, quelques mesures de bon sens s'imposent ! Affectez un numéro SDA qui ne correspond pas au numérotage de votre infrastructure connue et que vous prendrez soin de placer sur liste rouge. Un changement régulier des mots de passe d'accès est aussi conseillé, car ces derniers sont le plus souvent triviaux.

Ce niveau de sécurité est améliorable à l'aide de modems spécifiques : « callback » qui, sur chaque appel, rappellent un numéro prédéterminé, ce qui barre l'accès aux personnes autres que le mainteneur, à chiffrement pour garantir la confidentialité des échanges et une authentification du correspondant plus robuste. De plus, certains modems [3] ajoutent une journalisation de l'ensemble des données transitant par le modem afin d'assurer une traçabilité des opérations de maintenance.

Même si des améliorations sont introduites, les autocommutateurs ont tendance à peu journaliser les opérations d'administration. Quand on compare les opérations classiques d'administration faites sur un autocommutateur aux opérations réalisées sur un système d'exploitation, on peut s'apercevoir que le niveau de journalisation n'est pas du tout équivalent. On peut prendre pour exemple la création d'un poste sur un autocommutateur (à rapprocher de l'ajout d'un utilisateur sur un système d'exploitation) ou la modification d'un droit d'un poste (à rapprocher du changement du niveau de privilège d'un utilisateur). Ce type d'opération est systématiquement journalisé sur un système d'exploitation correctement sécurisé, ce qui n'est pas forcément le cas sur les autocommutateurs qui journalisent en priorité les événements liés à la qualité de service (incident sur les lignes, etc.).

Le réseau TCP/IP

Qu'il s'agisse du réseau d'administration ou des communications inter-commutateurs, les réseaux TCP/IP sont très souvent utilisés. La littérature MISCienne (en particulier sur les tests d'intrusions) regorge d'informations sur l'analyse de ce type de réseau. Il faut cependant noter que bien souvent, des services peu sécurisés (FTP, Telnet, etc.) sont utilisés avec des mots de passe triviaux sur les réseaux d'autocommutateurs [4].

Le PABX peut être utilisé suivant plusieurs objectifs. Un pirate ayant gagné un accès par la voie de télémaintenance tentera de rebondir sur d'éventuels réseaux joignables depuis le PABX. Ces interconnexions sont de plus en plus fréquentes, pour le partage d'annuaire (téléphonie, messagerie électronique), par exemple. La convergence « voix/données » ne semble pas indiquer que la vapeur va s'inverser.

Dans une autre optique, un pirate peut profiter de ces interconnexions pour s'introduire sur le PABX lui-même depuis un réseau interconnecté. Il tentera alors de modifier la programmation de ce dernier afin de mettre une fraude en place.

Les autres protocoles d'accès au PABX

Chacun des autres protocoles de communication du PABX est susceptible de contenir des vulnérabilités, mais leur mise en œuvre demande une forte spécialisation de l'attaquant, voire aussi des moyens importants.

Les connexions modems

Une angoisse fréquente des gestionnaires de PABX concerne l'existence de connexions modems non contrôlées qui mettent à mal la défense périmétrique offerte par les firewalls. Le Graal serait, un peu comme les sondes de « prévention » d'intrusion (et non de

d'utilisateurs et que beaucoup d'entre eux ont souvent les mêmes besoins, il serait fastidieux pour un administrateur de gérer l'attribution individuelle de ces droits ainsi que le suivi efficace de ces paramètres.

Du coup, cette attribution de droits se gère par l'intermédiaire de plusieurs tables liées qui constituent au final des profils d'utilisateurs. Nous allons reproduire les différentes étapes de configuration permettant de définir les droits d'un poste.

Tout d'abord, l'administrateur regroupe des numéros de téléphone suivant des critères de son choix. En général, comme les factures de téléphone visent à être optimisées, le principal critère est d'effectuer un regroupement par coût unitaire. Ces groupes de numéros homogènes constituent des zones de discrimination où se retrouvent classiquement les fameuses zones local/national/international/portables/numéro surtaxés. Ces regroupements doivent être soigneusement effectués et quelques erreurs classiques évitées. Par exemple, les numéros d'urgences devraient être regroupés dans une catégorie accessible à tous tandis que les préfixes correspondants aux FAI (0 860) devraient n'être pas référencés ou isolés dans une catégorie attribuée au compte-goutte. Ensuite, on crée des profils regroupant ces zones de discrimination, appelés catégories d'accès au réseau public. Ainsi, on pourra créer une de ces catégorie d'accès au réseau public ne regroupant que les zones de discrimination d'urgence et locale afin d'être attribuée ultérieurement à des stagiaires, et une seconde catégorie d'accès au réseau public constituée d'un plus grand nombre de zones de discrimination (local, portables, national et international) pour des commerciaux ayant des clients à l'étranger. Afin de limiter les abus en dehors des heures ouvrées, le contenu d'une catégorie d'accès au réseau public en termes de zones de discrimination peut varier suivant un facteur jour/nuit.

Enfin, chaque poste téléphonique (et par extension chaque usager) se voit affecter une des catégories d'accès au réseau public définies précédemment par l'administrateur, suivant ses besoins. Lors de l'analyse de sécurité, il faut impérativement identifier les zones de discrimination dangereuses puis remonter jusqu'aux utilisateurs de ces profils dangereux.

Les numéros abrégés

Les numéros abrégés permettent de définir des « alias » sur des numéros de téléphone. À titre d'exemple, tous les postes internes sont, par commodité, joignables par leurs derniers chiffres. Dans une entité possédant des satellites à travers le pays, voire le monde ou des interlocuteurs privilégiés, la numérotation abrégée peut être utilisée pour diminuer les besoins en termes de droit d'accès. En effet, lors de la composition d'un numéro, le PABX regarde d'abord s'il fait partie des numéros abrégés et, dans le cas contraire, vérifie que l'appelant a le droit d'accéder à ce numéro en regardant sa catégorie d'accès au réseau public.

En créant des numéros abrégés adéquats (« alias » vers certains numéros à l'international, de clients, etc.), il est possible de n'attribuer qu'un minimum de droits aux utilisateurs. Néanmoins, il est important de vérifier périodiquement si les numéros abrégés n'auraient pas été modifiés par une personne malveillante désirant augmenter discrètement ses droits.

La taxation

Contrairement aux particuliers, les contrats avec l'opérateur téléphonique ne comprennent pas l'émission d'une facturation détaillée. L'opérateur fournit en effet une facture globale qui ne vous permet pas de détecter les fraudes ou abus qui pourraient avoir lieu. Cependant, en cas d'activité manifestement inhabituelle, il se peut que l'opérateur vous signale ces anomalies, si lui-même s'en rend compte.

Pour traquer ce genre de dérives et plus généralement pour optimiser les coûts, les PABX intègrent des outils destinés à effectuer le suivi de la facturation. La configuration d'un tel outil est affaire de spécialiste, car il faut souvent jongler avec les différents modes de facturation d'un, voire plusieurs opérateurs. Les informations de taxation sont fournies par le PABX à une application externe, appelée console de taxation. Comme le PABX est un système temps-réel donnant la priorité à l'établissement et au maintien des canaux de communications voix, il se peut que les tâches relatives à la taxation se fassent oublier. Ainsi, certains tickets sont parfois perdus.

Tous ces aléas font qu'il n'est pas rare de voir des écarts de 10% entre les relevés fournis par l'opérateur et la taxation interne. Un écart trop important doit vous faire suspecter une fraude potentielle ou un sous-dimensionnement (perte de tickets de taxation) de votre PABX.

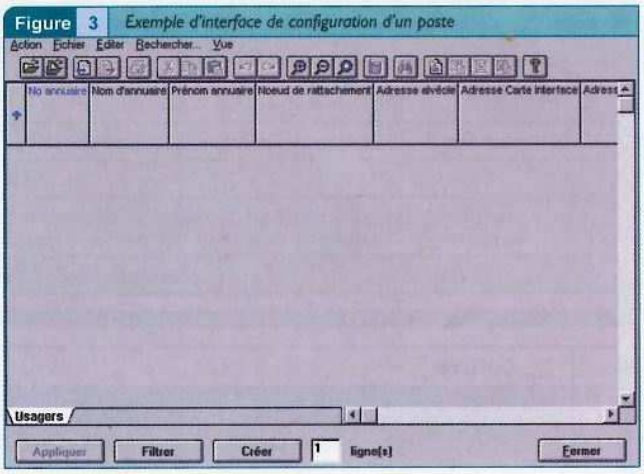
Les postes fictifs

Un poste fictif est un poste qui n'existe pas réellement, c'est-à-dire qu'il s'agit d'un poste virtuel. Ainsi, lorsqu'on appelle ce poste, aucun combiné ne sonne. Ce mécanisme peut être utilisé lors du changement de la numérotation de certains utilisateurs. Leur ancien numéro est « fictif » et amène à un message enregistré indiquant le nouveau numéro.

Ce type de poste représente un intérêt pour les attaquants, car lors de la mise en place d'une fraude sur le poste d'un utilisateur réel, celui-ci pourra la découvrir et/ou la désactiver (ex : mise en place d'un renvoi sur un poste). En revanche, si aucun combiné n'est associé au poste, aucun utilisateur ne détectera la fraude.

Il s'agit donc de tenir un inventaire des postes fictifs et de surveiller régulièrement l'apparition « spontanée » de ce type de poste.

Figure 3 *Exemple d'interface de configuration d'un poste*



En pratique, chaque poste téléphonique est identifié de manière unique dans l'interface de configuration du PABX à l'aide d'une adresse qui est la réunion, en principe, de l'identifiant du PABX de raccordement, de l'identifiant de la carte de raccordement dans le châssis du PABX et de l'identifiant du connecteur (branchement de la paire téléphonique) de raccordement sur cette carte (cf figure 3 page précédente). Un poste fictif est caractérisé par l'absence de ces paramètres ou l'usage de valeurs arbitraires (255 par ex.).

La justification d'un poste

Par défaut, tout poste déclaré apparaît dans les relevés de taxation effectués par le PABX. Pour rendre invisible un poste de la taxation, et donc masquer une fraude, il doit posséder la propriété de « non justification ». Imaginez l'intérêt que représente un poste fictif non justifié : toute fraude devient quasiment invisible s'il n'est pas effectué une recherche régulière de ce type de poste.

Étant donné la dangerosité d'une telle fonction, un inventaire des postes possédant cette propriété devrait tendre vers zéro. Il n'existe pas de réel moyen de convaincre un adepte de la non-justification à renoncer à son droit, donc à vous de jouer de vos atouts de persuasion (ou d'autorité) afin d'atteindre votre objectif. N'oubliez pas que le traitement des informations de taxation est soumis au secret professionnel [5] qui, s'il est respecté, remplit le même objectif que la non-justification.

Les fonctionnalités importantes

Les besoins en termes de sécurité sont finalement les mêmes que pour un système informatique classique : imputabilité des actions, confidentialité des échanges, etc.

De la même manière que pour la classification des droits d'accès au réseau public, il est facile de constater que la fonction d'un utilisateur dans l'entité détermine les fonctionnalités dont il a besoin.

Une grande partie des paramètres de fonctions sont donc regroupés dans des catégories d'exploitation (ou classes de facilité) qui sont ensuite attribuées aux postes utilisateurs. A vous d'identifier vos catégories les plus dangereuses et à les surveiller de près.

La substitution

Parmi les fonctions attribuables, certaines mettent en jeu plusieurs postes téléphoniques. C'est le cas de la substitution par exemple, qui permet depuis n'importe quel poste interne de récupérer les droits d'un poste particulier, après authentification par un code PIN bien évidemment.

Pour ce type de fonction, il est nécessaire de positionner deux notions de droit : la possibilité d'effectuer l'action pour un poste, ainsi que celle d'être « victime » de l'action. Suivant la nature de la fonction, les deux paramètres devront être sérieusement contrôlés, mais plus particulièrement le second. Il n'est pas rare de voir une attribution générale du droit d'effectuer des substitutions sur un autre poste, alors que le droit d'être substitué devrait être plus rare. Le principal

Plan de numérotation

	0	1	2	3	4	5
0	PRISE DIRECTE RESEAU RTCP					
1	Appel Opératrice	Annuaire Interne	Réponse au service réduit	Dérangement Interne	Libre	Dérangement Data
1#0	Consultation alarmes	Écoute guides vocaux	Tests Q23	Tests des tonalités	Préfixes ACD	Numérotation Z derrière UA
2	Libre	Libre	Consultation appel en attente	Renvoi Mevo simple	Libre	Annulation tous renvoi
20	Renvoi variable sur occupation	Renvoi immédiat	Accès Opérateur Alternatif sur non réponse	Renvoi temporisé ou non réponse	Renvoi sur occup	Modification poste associé
3	Libre	Libre	Libre	Libre	Libre	Num. abrégée Lille
4	Numérotation interne					
5	Numérotation interne					
6	Interception d'appel	Entrée Groupement numéro mémorisé	Parcage	Sortie Groupement	Réémission	Rappel du dernier appelant
7	Cadenas	Modification Code secret	Consultation Mevo	Rappel Rdv	Annulation rappel RdV	Garde commune
8	Libre					
9	Recherche de personne	Réponse à recherche de personne	RV immédiat/ recherche de personne	Renvoi différé sur recherche de personne	Libre	Libre
*	Surnumérotation Q23 (DTMF)					

Suffixe

0	1	2	3	4	5
	Va et vient	Double appel	Conférence	Entrée en tiers	Identification appel malveillant

risque associé est que cette fonction permet une élévation de privilège en se substituant à un poste à haut niveau de droits.

De plus, la taxation permet quelquefois de ne suivre qu'un niveau de substitution. Ainsi, en chaînant les substitutions, il devient possible de masquer complètement l'origine de l'appel.

Les renvois vers numéros externes

La technique la plus classique pour détourner des ressources en interne est la suivante : un renvoi d'un poste sur un numéro international ou portable est effectué par un employé avant de partir, puis celui-ci appelle, de l'extérieur, le poste interne qui sera ainsi renvoyé directement vers le numéro positionné. Ainsi, les communications seront aux frais de l'employeur.

Elle peut cependant devenir plus subtile pour un attaquant externe motivé. Imaginez que cet attaquant puisse mettre en œuvre, à distance (par la DISA par exemple) un renvoi sur un poste. Dès le retour de l'utilisateur réel du poste, le renvoi sera désactivé car il ne pourrait pas utiliser son poste.

L'attaquant, s'il a des privilèges élevés sur l'autocommutateur, mettra en œuvre un renvoi sur un poste fictif non justifié qu'il aura créé.

La parade consiste à ne pas autoriser ce type de renvoi et/ou faire la chasse aux postes pourvus de ce droit sans oublier de se méfier des numérotations abrégées qui peuvent entraîner une taxation coûteuse. En effet, les renvois internes sont toujours autorisés et les numéros abrégés font partie du domaine interne.

Le plan de numérotation

Pour l'usager, l'utilisation d'une fonctionnalité spécifique passe par la composition d'un code correspondant à la fonction : un préfixe ou un suffixe pour les fonctionnalités accessibles en cours de communication. Parmi les préfixes connus d'une grande majorité des usagers d'un PABX, il y a le 0 (ou autre chiffre) qui vous permet de communiquer sur le réseau public (« prise de faisceau externe »). En rassemblant les données de la table des préfixes et celle des suffixes, un plan de numérotation est construit. Le tableau ci-dessous présente un exemple de plan de numérotation avec en première colonne la première touche composée suivie de la seconde donnée en première ligne.

Une lecture rapide montre que les numéros internes commencent par 4 ou 5. S'il s'agit d'une numérotation sur quatre chiffres, les postes internes sont donc compris entre 4000 et 5999. Une numérotation abrégée (préfixe 35-39) pointe vers des sites dont certains semblent à l'étranger. Ceci indique qu'il existe une interconnexion de PABX nationale, voire internationale, qu'un attaquant pourra tenter d'exploiter. Un certain nombre de fonctions (mise hors service, consultations alarmes), bien qu'accessibles à tous, ne sont certainement pas portées à la connaissance de tous les usagers.

Bien qu'un plan de numérotation soit entièrement personnalisable, les administrateurs n'effectuent souvent qu'un nombre limité de retouches par rapport au modèle standard du PABX. Pour cette raison, connaissant le modèle du PABX utilisé, il est possible pour un attaquant de deviner les préfixes associés aux fonctionnalités dangereuses (voir tableau ci-dessous).

6	7	8	9	*	#
Libre	Libre	Sécurité PSI	Libre	Libre	Libre
Accès MEVO	Mise en/hors service poste sur occupation	Débort. associé non réponse	Débort. associé /	Rondier	Libre
Annulation renvoi fixe par destinataire	Libre	Libre	Libre	Sélection ligne principale	Sélection lignes secondaires
Validation débordement	Annulation débordement	Renvoi immédiat à distance	Annul renv imm à distance	Libre	Libre
Num. abrégée Lyon	Num. abrégée Toulouse	Num. abrégée Bordeaux	Num. abrégée Partenaires	Libre	Libre
Appel rapide de l'associé	Conférence programmée en transp. Q23	Interception d'appel dirigé	Annulation RASPO de ligne	Passage auto	Protection
Libre	Substitution	Libre	Libre	Appel malveillant	Secret d'identité
Libre	Libre	Adressage système	Libre	Libre	Modification langue

6	7	8	9	*	#
62 : Parcage	Attente sur occupation	Débordement MEVO	Débordement sur recherche	Sur-num. Q23	

Un peu de ménage dans les fonctionnalités

Parmi ce plan de numérotation, on trouve un certain nombre de fonctions identifiées comme dangereuses et peut-être inutiles dans l'exemple présenté : préfixe de substitution et suffixe d'entrée en tiers. Trois méthodes sont possibles pour en interdire l'usage : supprimer sur chaque poste ou catégorie d'exploitation les droits relatifs à cette fonction (peut être lourd et fastidieux), supprimer le préfixe/suffixe permettant d'accéder à cette fonction (simple, rapide, mais nécessite de vérifier périodiquement que quelqu'un n'a pas modifié le paramétrage) et enfin la meilleure solution qui est l'association des deux précédentes...

Les sonneries

Encore une preuve que la sécurité des PABX est une affaire de bon sens, les sonneries sont un élément de sécurité. En effet, sur les postes analogiques (ou numériques avec un écran cassé), les fonctionnalités intrusives lors d'une communication (appel en attente, entrée en tiers, etc.) sont uniquement annoncées par des signaux Y sonores.

Ces signaux sont déclarés sous la forme d'une succession de couples (valeur du signal, durée du signal) qui caractérisent une sonnerie. Une des manières de supprimer tout avertissement lors d'une fonction intrusive est de modifier les sonneries pour que chacun des couples ait une durée nulle, la sonnerie étant alors inaudible.

Il faut donc surveiller régulièrement d'éventuelles modifications de ces paramètres. Il est préférable d'associer aux fonctionnalités les plus dangereuses des sonneries très caractéristiques et de sensibiliser les utilisateurs à ce fait de manière à ce qu'ils ne confondent pas une entrée en tiers avec un signal d'appel.

Les sonneries sont aussi un bon moyen de « fingerprinter » l'autocommutateur utilisé par une entité. En effet, les sonneries sont propres à chaque constructeur, voire à chaque gamme et un spécialiste peut donc, juste à l'écoute d'une sonnerie, identifier très précisément votre PABX.

Conclusion

J'espère que cet article vous a apporté un certain nombre d'informations, même si le sujet est loin d'être complètement traité. Malheureusement, la littérature publique (forums **fr.reseaux.telecoms.pabx**, publications, etc.) concernant ce domaine est faible. Ceci ne signifie nullement qu'il n'existe pas de problèmes de sécurité, et de solutions permettant d'y remédier comme en témoigne le document du NIST [6]. N'hésitez donc pas à lire attentivement les documentations accompagnant vos équipements afin de bien juger de la dangerosité des fonctions que vous souhaitez activer.

Merci à Céline de me supporter.

Bibliographie

[1] Hadi El-Khoury, MISC 11, *Tests intrusifs sur les infrastructures télécom d'entreprises*

[2] CLUSIF-ERCOM, *Malveillance téléphonique*
https://www.clusif.asso.fr/fr/production/ouvrages/pdf/MalvTel_Infosecurity031128.pdf

[3] Modem Safemodem
http://www.ercom.fr/siteweb/catalogue/fiche.php?id=12

[4] Avis CERTA
http://www.certa.ssi.gouv.fr/site/CERTA-2002-ALE-005/index.html.2.html

[5] CNIL, *Les autocommutateurs téléphoniques sur le lieu de travail*
http://www.cnil.fr/index.php?id=1588

CNIL Norme simplifiée N°40
http://www.cnil.fr/index.php?id=1244

[6] Richard Kuhn (NIST), *PBX Vulnerability Analysis*
http://csrc.nist.gov/publications/nistpubs/800-24/sp800-24pbx.pdf

Glossaire

DISA	« *Direct Inward System Access* » ou accès direct aux services internes.
LIA	Liaison Inter-Automatique. Lien privé destiné à relier deux autocommutateurs et assurer la transmission de la signalisation entre ces équipements suivant des protocoles spécifiques (propriétaires ou standards).
MEVO	MEssagerie VOcale. Équipement dédié hébergeant les boîtes vocales des utilisateurs d'un PABX.
MIC	Modulation par Impulsions et Codage. Par extension, nom donné au lien numérique standard (2Mbps - 30 canaux voix + 2 canaux de signalisation) fourni par les opérateurs pour l'interconnexion au réseau public.
PABX, PBX	*Private (Automatic) Branch eXchange*, autocommutateur téléphonique.
RNIS	Réseau Numérique à Intégration de Services (ISDN).
RTCP	Réseau Téléphonique Commuté Public. En France, le réseau public est entièrement numérique à l'exception du lien reliant l'abonné à l'opérateur qui est analogique (sauf ligne RNIS et connexions de PABX).
SDA	Sélection Directe à l'Arrivée. Un numéro SDA est le numéro affecté à un poste sur le réseau public. En France, il est de la forme ZABPQMCDU (numéro à 9 chiffres précédé du préfixe de l'opérateur). Les administrateurs configurent les PABX pour que le numéro interne d'un poste corresponde au MCDU (voire QMCDU sur des réseaux importants) du numéro SDA. Un poste interne n'est pas obligatoirement associé à un numéro SDA, il n'est alors pas joignable directement depuis le réseau public.
Sur-numérotation Q.23	Méthode de signalisation permettant, par exemple, à un usager d'un poste analogique d'interagir avec un serveur vocal. Chaque appui sur une touche génère un signal sonore DTMF (*Dual-Tone Multi-Frequency*) interprété par le serveur vocal distant.

L'entrée en tiers ou « Quelqu'un serait-il à l'écoute » ?

Hadi El-Khoury
h.elkhoury@miscmag.com
Consultant en Sécurité des Systèmes d'Information

Les idées présentées ci-dessous constituent des pistes de réflexion visant à sensibiliser le lecteur sur les risques induits par la convergence téléphonie informatique et la nécessité absolue de garantir une coordination minimale entre les équipes informatiques et téléphoniques. Elles ne constituent en aucun cas une incitation à des malveillances de quelque nature que ce soit, d'ailleurs lourdement réprimées par la Loi.

Pour faire suite au premier article traitant des risques pesant sur les infrastructures téléphoniques d'entreprise [1] et compléter l'article sur le durcissement des PABX de ce même dossier, est présenté ci-dessous un scénario d'attaque visant à s'introduire sur une communication téléphonique déjà établie.

Bien qu'appartenant à la famille « piratage des fonctionnalités intrinsèques au PABX », ce scénario nécessite la combinaison de plusieurs autres manipulations puisées dans une deuxième famille de vulnérabilités (notamment celle du piratage par les auxiliaires du PABX).

Ainsi, la fonctionnalité « entrée en tiers » est détournée de manière malveillante pour écouter une conversation établie entre deux victimes, et ce à leur insu : à savoir en désactivant tout « BIP » (imposé par la « Loi Informatique et Libertés ») susceptible de dévoiler la présence de l'intrus.

Cette attaque se décline en plusieurs étapes et nécessite des connaissances en administration de la téléphonie.

Par suite, l'article respectera la démarche suivante :

- → Fonctionnalité ciblée : entrée en tiers
- → Accès en administrateur à la console de gestion du PABX
- → Modification des catégories d'exploitation téléphonique
- → Accès à la session courante des Postes Opérateurs (standardistes téléphoniques)
- → Intrusion proprement dite sur la communication établie
- → Variante possible : élévation de privilèges par substitution à un abonné privilégié
- → Moyens de protection, meilleures pratiques, surveillance

La fonctionnalité ciblée : entrée en tiers

La documentation officielle d'Alcatel [3] donne de la fonctionnalité « entrée en tiers » la définition suivante : *L'entrée en tiers permet à un usager d'intervenir dans une communication entre deux postes sans y avoir été invité* (le motif « légitime » est, en tant que tel, discutable).

Pendant la durée de l'entrée en tiers, les trois usagers forment une conférence à trois.

Ce service, généralement réservé aux opératrices :

- → est accompagné d'un bip émis tout au long de l'entrée en tiers (tonalité numéro 24),
- → ne peut pas être mis en œuvre sur les postes disposant d'une protection contre l'entrée en tiers.

A est le poste intervenant dans la conversation de B et C.

1. A appelle B, déjà en conversation avec C.
2. A effectue une entrée en tiers dans la conversation de B et C.

Accès en administrateur à la console de gestion du PABX

Les PABX actuels appartiennent simultanément aux infrastructures téléphoniques et informatiques de l'entreprise. Autrement dit, mis à part leur rôle téléphonique classique, les PABX disposent d'une ou de plusieurs adresses IP et proposent des services TCP/IP usuels comme Telnet, FTP, etc. permettant une gestion de l'autocommutateur téléphonique en ligne de commande.

Par ailleurs, certains constructeurs ont développé des interfaces de gestion très ergonomiques rendant intuitive l'administration de ces équipements (Alcatel OmniVista 4760, Matra 7430, etc.). Soucieux de faciliter au maximum l'accès à ces consoles, ceux-ci les ont dotées d'une interface Web, affranchissant l'administrateur téléphonique de l'utilisation du client de connexion correspondant. L'expérience montre que la visibilité de cette interface Web (figure 1) n'a rien à envier à celle des adresses IP attribuées aux auxiliaires de la téléphonie (gestion, taxation, Postes Opérateurs, etc.) !

Figure 1

Cependant, ces équipements, très souvent installés par un intégrateur, ne font l'objet d'aucune sécurisation particulière. Hormis les « sacro-saints mots de passe constructeur » jalousement gardés par les intégrateurs, les mots de passe par défaut, déjà publiés

en partie par le CERTA [2], (que ce soit au niveau du système d'exploitation sous-jacent ou de la console de gestion) ne sont pas changés. Les adresses IP attribuées ne sont pas regroupées au sein d'un sous-réseau cloisonné. Il est alors courant de trouver des PABX visibles à partir de n'importe quel segment du système d'information.

Modification des catégories d'exploitation téléphonique

En combinant les précédentes manipulations avec la consultation de l'annuaire téléphonique de l'entreprise (plus ou moins sophistiqué et faisant partie de la suite logicielle du PABX), il est possible de connaître la catégorie d'exploitation téléphonique de la victime. Muni de cette information précieuse, l'attaquant potentiel se reporte à la console d'administration du PABX et y apporte les modifications nécessaires (voir Figure 2).

En effet, pour mettre en œuvre cette fonctionnalité, il faut, au niveau de la console d'administration :

→ Donner les droits d'exploitation aux postes concernés.

→ S'assurer que les postes concernés ne sont protégés ni contre les entrées en tiers, ni contre les bips.

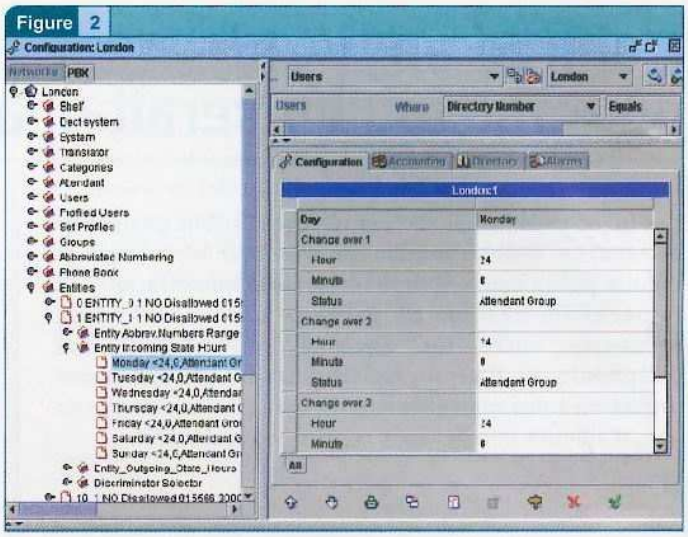
→ Définir le suffixe d'activation du service.

Autorisation et protections

L'autorisation d'exploitation, comme les protections possibles, se définissent dans la catégorie d'exploitation téléphonique associée au poste.

Attributs	
N° Catégories d'expl. tél	Entrer le numéro de la catégorie d'exploitation téléphonique
Suffixes	
Entrée en tiers	0 : l'usager n'a pas le droit d'effectuer une entrée en tiers.
Droits	
Prot. contre toute entr. tiers	0 : l'usager ne dispose pas d'une protection contre toute entrée en tiers.
	1 : l'usager dispose d'une protection permanente contre les entrées en tiers pour toutes ses communications, même si l'appel en tiers est prioritaire.
Prot. contre entr. tiers poste	0 : l'usager ne dispose pas d'une protection contre les entrées en tiers.
	1 : l'usager dispose d'une protection permanente contre les entrées en tiers pour toutes ses communications, sauf si l'appel en tiers est prioritaire.
Protégé contre les bips	0 : l'usager ne dispose pas d'une protection contre les bips.
	1 : l'usager dispose d'une protection contre les bips et ne peut donc pas subir d'entrée en tiers.

Figure 2



Définition du suffixe

L'activation de la fonction d'entrée en tiers se fait par composition d'un suffixe.

Nom de l'objet : **Traducteur > Plan des suffixes**

Attributs	
Numéro	Entrer le numéro du suffixe. Ce numéro doit être compatible avec le plan des suffixes de l'installation
Exploitation du suffixe	Sélectionner : Entrée en tiers

Accès à la session courante des Postes Opérateurs (standardistes téléphoniques)

Il est à noter que les Postes Opérateurs sont des cibles intéressantes dès lors qu'une prise de main distante de la session courante (via la connexion à un VNC « oublié » ou autre) permet une observation du trafic téléphonique entrant. Il faut également souligner que certaines suites logicielles pour Postes Opérateurs permettent un accès en VT100 à la console du PABX ! (Voir Figure 3). Ceci constitue une précieuse collecte d'informations permettant de mieux cerner les profils appelants et/ou appelés et de choisir plus judicieusement les conversations à écouter.

L'intrusion proprement dite sur la communication établie

Ainsi, il ne reste plus qu'à effectuer l'entrée en tiers proprement dite. Celle-ci ne pouvant se faire que sur poste local occupé, en conversation avec un autre poste (local ou externe).

1 Composer le numéro du poste à joindre.

Si les conditions d'autorisation et de protection sont bien remplies, un guide vocal indiquera le suffixe à composer pour une entrée en tiers.

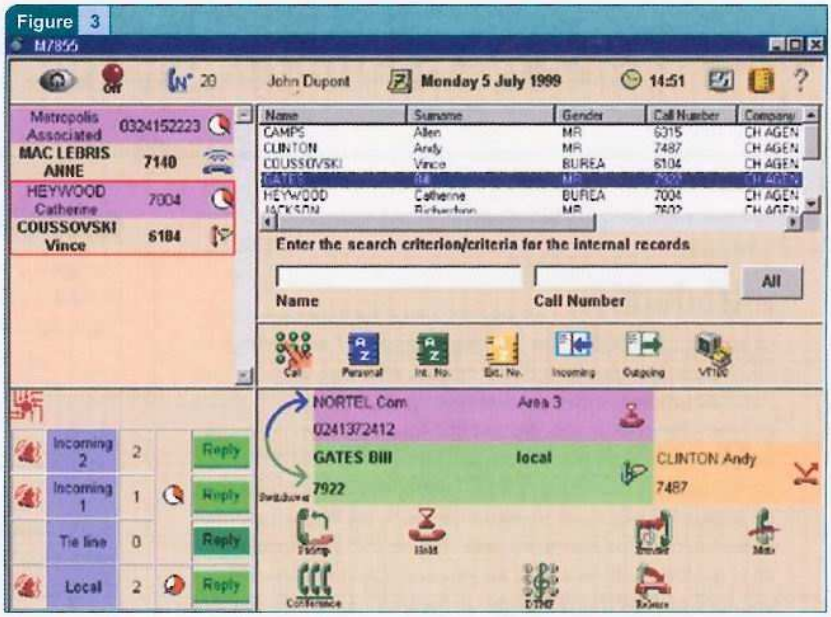
2 Composer le suffixe d'entrée en tiers ou, sur un poste à touches de fonctions dynamiques, appuyer sur la touche Intrus.

Les trois postes sont en conférence à trois. Un bip sonore, s'il n'a pas été supprimé, répété tout au long de l'intrusion, avertit les deux usagers de la présence d'un autre poste dans la communication.

3 Pour mettre fin à l'intrusion dans cette conversation, il suffit de raccrocher.

Si l'un des deux correspondants raccroche pendant l'intrusion, le poste intrus est également déconnecté.

Figure 3



Variante possible : élévation de privilèges par substitution à un abonné privilégié

La notion d'élévation de privilèges, largement connue dans le domaine des systèmes d'exploitation, est également présente en téléphonie par le biais de la fonction « récupération de fonctionnalités » ou « substitution » [voir Définitions ci-contre].

Ainsi, pour s'affranchir de toute ou partie de la modification des protections/autorisations relatives à l'entrée en tiers, il suffit de repérer les abonnés téléphoniques affectés aux catégories d'exploitation téléphonique les plus permissives (à savoir les abonnés les plus privilégiés : cadres dirigeants, standardistes, etc.). Il s'agit ensuite de détourner la fonction « récupération des fonctionnalités » de son rôle initial pour tenter de récupérer à partir d'un poste téléphonique banalisé, à savoir non privilégié, les fonctionnalités d'un poste privilégié et de disposer de toutes les permissions qui lui sont associées. C'est ainsi que se matérialiserait l'élévation de privilèges.

Les étapes nécessaires à la mise en application de cette attaque suivent le cheminement suivant :

→ Accéder à la console d'administration.

→ Vérifier les droits liés à la substitution définis dans la catégorie d'exploitation téléphonique associée au poste de l'attaquant.

→ Y apporter les modifications souhaitées au niveau des droits, des exploitations du poste, etc.

→ Relever ou définir les préfixes d'activation de la substitution, sachant que celle-ci pourrait être partielle ou totale selon le cas.

→ Décrocher, attendre la tonalité, composer le préfixe, entrer le numéro du poste de la victime privilégiée, entrer le code personnel de ce poste (qui a été relevé ou redéfini).

→ La substitution est alors effectuée …

Moyens de protection, meilleures pratiques, surveillance

Les fonctionnalités d'un PABX semblent intarissables (ainsi que les scenarii d'attaques qui en découlent) et leur bonne gestion, quoique complexe, est indispensable à la bonne application d'une politique de sécurité cohérente et homogène sur l'ensemble du système d'information.

D'où l'extrême nécessité de documenter le paramétrage des autocommutateurs, les affectations des utilisateurs à telle ou telle catégorie d'exploitation téléphonique, et surtout d'être à même de surveiller les actions entreprises sur le PABX et ses serveurs auxiliaires, bien sûr après les avoir correctement protégés (se reporter à l'article « Durcissement du PABX »).

Définitions

Catégorie d'exploitation téléphonique :
Ensemble de paramètres, droits, attributions, protections dont dispose un abonné ou un groupe d'abonnés.

Poste Opérateur sur PC :
Le rôle d'un poste opérateur est l'accueil des appels extérieurs et leur acheminement vers les postes de l'installation. Il peut être rendu plus ergonomique d'utilisation grâce à une application sur PC.

Substitution ou *squatt* : (définition donnée par Matra, [4])
Les abonnés utilisent les facultés de leur téléphone personnel sur n'importe quel téléphone de l'entreprise quelle que soit la catégorie ou l'état du terminal. La substitution est appliquée chaque fois que les utilisateurs effectuent un appel et nécessite un code confidentiel, utilisé pour restreindre cette fonction aux personnes autorisées.

Références

[1] Hadi El-Khoury, *« Tests Intrusifs sur les Infrastructures Télécoms d'Entreprise »*, MISC 11, janvier - février 2004

[2] Risque de compromission des autocommutateurs (PABX) ALCATEL 4400
http://www.certa.ssi.gouv.fr/site/CERTA-2002-ALE-005/index.html.2.html

[3] Alcatel 4400, documentation système

[4] Matra 6500, documentation système

Sécurité de la Voix sur IP

Introduction

La voix sur IP est déjà, ou va devenir, un projet stratégique en 2005 pour bon nombre d'entreprises et d'opérateurs. Pour l'utilisateur, elle permet uniquement de téléphoner à moindre coût via l'Internet, la voix restant une forme de communication bien plus conviviale que le courrier électronique ou les formes de messageries instantanées. Pour l'entreprise et les opérateurs, ce facteur « coût de la communication » est important, mais le déploiement de réseaux privés virtuels MPLS, l'introduction de la qualité de service dans les réseaux (QoS), la convergence voix-données (CTI), les divers projets de consolidation des deux dernières années, l'arrivée des auto-commutateurs IP, la disponibilité de postes téléphoniques intégrant des fonctionnalités de plus en plus avancées sont des facteurs tout au moins aussi déterminants. Des études récentes montrent que la sécurité de la VoIP est un élément clé pour les décideurs, mais les déploiements observés ont malheureusement tendance à montrer le contraire.

Après avoir présenté les principaux protocoles liés à la voix sur IP puis les rudiments de la sécurisation des différents équipements, nous détaillerons plusieurs attaques et quels « ajouts » sécurité en limitent l'impact. Pour finir, nous discuterons de l'interception de trafic et introduirons deux évolutions récentes de la VoIP.

Les différents protocoles

Nous allons nous intéresser tout particulièrement aux solutions de voix sur IP qui reposent sur les protocoles SIP [2] (signalisation et contrôle) et RTP [12] (transport de la voix). Il nous semble important de rappeler qu'une solution complète repose sur un large nombre de protocoles et que H.323, qui n'est ici que cité, est encore présent dans bon nombre de déploiements.

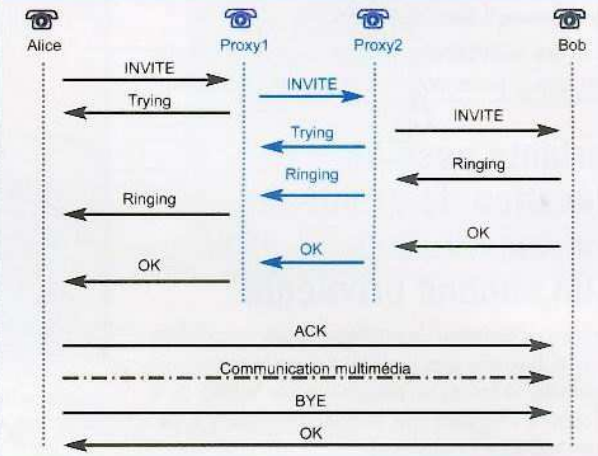
SIP

SIP (*Session Initiation Protocol*) est le standard IETF pour la signalisation (établissement, terminaison, redirection, relayage, etc.) de communications multimédias interactives. Ce protocole est de nos jours celui qui est déployé couramment. Le format est proche d'une adresse de messagerie : **sip:nico@securite.org** avec une syntaxe proche de celle de HTTP.

SIMPLE (*SIP for Instant Messaging and Presence Leveraging Extensions*) est une extension de SIP dont le but est de supporter les messageries instantanées.

Le projet PROTOS [16] s'est intéressé à SIP, et comme pour SNMP, a trouvé un nombre important d'implémentations qui ne passaient pas le « batch » de tests sans se planter ou redémarrer. Cela concerne aussi bien les téléphones que les relais SIP et les équipements de sécurité.

Figure 1 *Exemple de session SIP*



Le « *SIP proxy* » joue le rôle de relais et fait suivre une requête SIP au prochain serveur. Le « *SIP redirect server* » renvoie une réponse au client, contenant le prochain serveur à contacter. Le « *SIP registrar* » enregistre le lien entre l'adresse IP et l'adresse SIP (figure 1).

Comme pour IPsec, des solutions alternatives ont dû être développées pour gérer les contraintes introduites par la traduction d'adresses (NAT).

Principales méthodes SIP

Méthode	Principe
INVITE	Débute une communication SIP. Création du *Call-ID* et éventuellement du *tag* sur le champ *From*.
REGISTER	Enregistre un utilisateur auprès d'un nœud VoIP (non utilisé en communication directe entre 2 clients).
ACK	Confirmation de l'établissement de la session SIP. *Call-ID* identique à celui du paquet INVITE associé. Ajout éventuel du *tag* sur le champ *To* du paquet INVITE relatif.
BYE	Met fin à la communication. Même *Call-ID* que les paquets précédents. Idem si utilisation des *tags*.
CANCEL	Annule un SIP INVITE (appel). Même *Call-ID* et éventuellement *tag* du champ *From* du SIP INVITE.

Pierre Betouin - **pierre.betouin@security-labs.org**, École supérieure d'informatique, d'électronique et d'automatique (ESIEA)
Patrick Chambet - **http://www.edelweb.fr** - **http://www.chambet.com** - **patrick@chambet.com**, Consultant Senior, EdelWeb - Groupe ON-X
Nicolas Fischbach - **nico@securite.org** - **http://www.securite.org**, Senior Manager, Network Engineering Security, COLT Telecom

H.323

H.323 [14] a été le premier protocole développé pour permettre des communications multimédias. SIP est son « concurrent ». H.323 est relativement complexe et SIP tente de simplifier les échanges en utilisant une sémantique proche de HTTP.

H.235 [13] définit certains mécanismes de sécurité (authentification et chiffrement).

Les protocoles secondaires

Le service DNS est utilisé pour fournir des services d'annuaire et de localisation. TFTP et HTTP sont utilisés par les téléphones et différents autres éléments pour télécharger leur configuration. ENUM permet de lier des adresses SIP via DNS aux numéros de téléphone au format E.164.

Sécurité des équipements

Les offres commerciales de VoIP nécessitent des serveurs et des applicatifs pour fonctionner. Nous allons prendre l'exemple d'une infrastructure Cisco et détailler les besoins de sécurité sur les différents équipements.

Serveurs Windows

Les applicatifs Cisco sont hébergés sur des serveurs Windows. Ces serveurs sont bien souvent installés par défaut et laissés en l'état. Il est donc nécessaire de commencer par sécuriser l'OS de ces serveurs (voir MISC 2 et [9]).

Les points suivants sont particulièrement à surveiller, notamment sur les serveurs pré-Windows 2003 :

- → filtrer les ports accessibles sur les serveurs depuis le réseau des utilisateurs, au niveau des routeurs ;
- → désinstaller ou stopper les services inutiles ;
- → mettre à jour le serveur avec les derniers correctifs de sécurité ;
- → interdire les connexions NetBIOS anonymes ;
- → appliquer des permissions d'accès strictes sur le système de fichiers et la base de registre ;
- → activer l'audit de sécurité et configurer des journaux de grande taille ;
- → installer IIS (nécessaire pour le Call Manager par exemple) de manière sécurisée (voir MISC 1 et [10]).

L'application Cisco Call Manager utilisant également une base de données SQL Server, il est très important de sécuriser particulièrement celui-ci.

Pour cela, les recommandations principales sont les suivantes :

- → appliquer les derniers correctifs de sécurité de SQL Server ;
- → faire tourner les services de SQL Server sous des comptes spécifiques, non administrateurs ;
- → attribuer un mot de passe robuste au compte "sa" ;
- → supprimer les bases par défaut ;
- → si possible (en fonction des applications), créer des utilisateurs spéciaux, voire nominatifs, et leur attribuer des mots de passe non triviaux ;
- → appliquer des permissions d'accès sur les objets de SQL Server (tables, procédures stockées, etc.) en fonction des utilisateurs ;
- → supprimer les procédures stockées par défaut si elles ne sont pas utilisées.

A noter que, sans sécurisation particulière, il est possible d'effectuer un transfert de zone DNS du domaine interne utilisé par défaut (**avvid.com**).

On y trouve également la liste des services et des ports utilisés par les applicatifs de gestion VoIP :

```
[[123.123.123.123]]
avvid.com.      SOA             unity01.avvid.com admin. (26 900 600 86400 3600)
avvid.com.      A               123.123.123.123
avvid.com.      NS              unity01.avvid.com
_kerberos._tcp.default-first-site-name._sites.dc._msdcs
SRV             priority=0, unity01.avvid.com
[...]
```

Il convient donc d'interdire le transfert de zone au niveau des serveurs DNS internes.

Serveurs *nix

En ce qui concerne la sécurisation des serveurs Unix, il est recommandé d'appliquer les recommandations générales suivantes, en fonction des applicatifs VoIP hébergés :

- → minimisation du système (désinstallation des démons et serveurs inutiles) ;
- → appliquer les derniers correctifs de sécurité de l'OS ;
- → créer des comptes spécifiques et leur attribuer des mots de passe robustes ;
- → *chrooter* (mise en « cage ») les applicatifs et les faire tourner sous des comptes non privilégiés ;
- → appliquer les derniers correctifs de sécurité des applicatifs ;
- → activer les logs (journalisation) au maximum.

Des informations sur la sécurisation d'Asterisk par exemple, un PBX logiciel tournant sur Unix, se trouvent ici : [17].

Téléphones

Le téléphone peut se présenter sous deux formes : soit un téléphone « classique », soit un téléphone « logiciel » qui s'exécute sur l'ordinateur de l'utilisateur. En terminologie SIP c'est un UA (*User Agent*).

Au niveau des téléphones IP, l'accès à la partie protégée par *HTTP Basic Authentication* se fait en clair :

```
http://IP-Phone/CGI/
```

Si le mot de passe est le même sur tous les téléphones, n'importe qui peut exécuter des commandes de configuration sur l'ensemble des téléphones.

Il est bien sûr recommandé de mettre à jour le *firmware* des téléphones IP, et de désactiver les interfaces de gestion du type HTTP et Telnet des téléphones IP (cela peut être fait depuis l'application Cisco Call Manager, que nous étudierons plus loin).

Signalons qu'il est possible d'effectuer un déni de service sur un téléphone IP par l'intermédiaire de l'application Cisco Call Manager 3.x. En effet, celle-ci gère le profil des utilisateurs, qui est téléchargé sur le téléphone IP lorsque l'utilisateur se logue. En dépassant largement la limite de 99 services téléphoniques IP par téléphone (en développant par exemple un script qui crée plus de 1000 services dans l'application Call Manager), le profil de l'utilisateur devient inutilisable sur un téléphone IP : on ne peut plus ni se loguer sur un téléphone, ni se déloguer, et celui-ci devient inutilisable (plus de tonalité). Il est nécessaire de l'éteindre et de le rallumer, après avoir corrigé le profil utilisateur dans le Call Manager.

Autres équipements

Certains constructeurs proposent des équipements propriétaires pour gérer la VoIP. Cisco propose par exemple les passerelles voix-données VG 200 et VG 248, fonctionnant sous IOS, le système d'exploitation des routeurs Cisco.

Il est fréquent de rencontrer le service Telnet ouvert sur de tels équipements. Il est donc fortement recommandé de ne pas laisser l'interface d'administration des équipements accessible depuis l'ensemble du réseau local.

De plus, étant donné les possibilités d'interception des connexions, même à travers le *switch* (cf. plus loin dans cet article), il est fortement déconseillé d'utiliser un protocole d'administration non chiffré.

Nous vous recommandons d'activer les fonctionnalités de sécurité présentes dans IOS et CatOS (voir MISC 1 ou [15]).

Applicatifs

Les applications de gestion du réseau VoIP sont bien souvent des applications Web, et, comme telles, elles sont sensibles aux attaques classiques contre ce type d'applications (voir Linux Mag HS No 12 et [11]). Il est donc nécessaire de prendre un soin particulier à leur sécurisation, d'autant plus que certaines d'entre elles ont besoin d'être accessibles depuis une grande partie du réseau local, voire depuis l'Internet !

Le Call Manager, par exemple, fournit des fonctions de base de gestion des appels, des utilisateurs et des configurations, mais également des fonctionnalités avancées comme la conférence, les boîtes vocales, etc. Il peut être vu comme un IP PBX. À ne pas confondre avec des PBX traditionnels (pas de VoIP) que l'on peut administrer à distance via une connexion TCP/IP (qui remplace la connexion locale ou de télé-maintenance via un modem). L'application Unity, quant à elle, est l'applicatif permettant d'avoir une messagerie vocale VoIP.

L'authentification lors des accès au Call Manager 3.x se fait par l'intermédiaire de formulaires HTML :

```
http://CallMgr/CCMUser/logon.asp
http://CallMgr/CCMAdmin/logon.asp
http://CallMgr/CCMCIP/authenticate.asp
http://CallMgr/ma/desktop/maLogin.jsp
```

De même, l'authentification pour l'accès à l'application *Cisco IP Manager Assistant* (IPMA) se fait à l'aide d'un formulaire HTML et d'une applet Java.

Dans tous les cas (accès utilisateurs et accès administrateur), le trafic réseau transite en clair. Il est facile, par une attaque ARP sur les commutateurs (voir plus loin), de capturer le trafic afin de récupérer le mot de passe de l'administrateur lorsqu'il se connecte. Il est donc indispensable de forcer l'utilisation de SSL pour accéder aux pages HTML d'authentification applicative.

D'autre part, les données saisies par l'utilisateur dans les formulaires HTML de l'application ne sont contrôlées que côté client (par des scripts JavaScript), ce qui est l'erreur classique : elles ne sont pas contrôlées à nouveau côté serveur.

Il est donc facile pour un attaquant d'outrepasser les contrôles afin d'entrer des données dangereuses dans l'application (code HTML, scripts, injection SQL, etc.).

Des applications directes de ce type de vulnérabilité sont par exemple les suivantes :

- → Déni de service sur les téléphones IP (voir plus haut) ;
- → *Cross Site Scripting* (XSS) : voir figure 2 ci-contre.

L'exemple ci-dessus montre qu'il est possible de récupérer le cookie d'authentification d'un utilisateur. Ce cookie est suffisant pour accéder à l'application à sa place.

D'autres recommandations, plus classiques, sont également à prendre en compte lors de l'installation des applications de gestion VoIP :

- → configurer le Call Manager, IIS et/ou TomCat afin de ne pas afficher des messages d'erreur détaillés pouvant fournir des informations précieuses à un attaquant ;
- → supprimer les répertoires par défaut et tous les répertoires et fichiers inutiles ;
- → installer un filtre d'URLs (de type URLScan) afin d'interdire les attaques classiques sur les URLs et les caractères spéciaux ;
- → etc. (voir [11]).

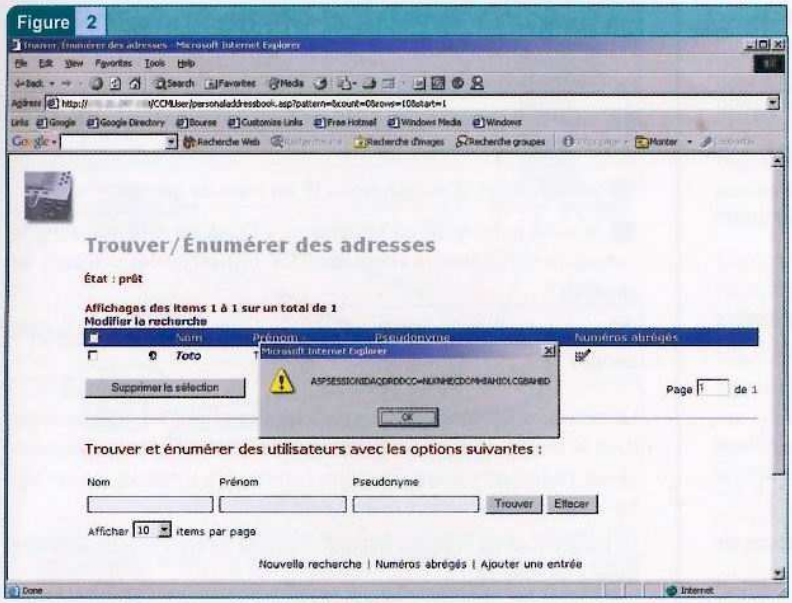
Il faut également savoir que l'accès à l'annuaire d'entreprise est en libre accès par défaut, aussi bien à partir des téléphones IP que de tout navigateur Web : il est possible de faire des recherches et de consulter l'ensemble de l'annuaire à l'aide de requêtes de la forme :

```
http://CallMgr/CCMCIP/xmldirectorylist.asp?l=A&f=&start=1
```

Le résultat donne les informations suivantes :

```
<CiscoIPPhoneDirectory>
<Prompt>Records 1 to 32 of 1254</Prompt>
-
<DirectoryEntry>
<Name>Glloq, Alain</Name>
<Telephone>12345</Telephone>
</DirectoryEntry>
[...]
</CiscoIPPhoneDirectory>
```

Figure 2



De même, un accès LDAP anonyme aux serveurs applicatifs révèle le contenu de l'annuaire :

```
ld = ldap_open(«123.123.123.123», 389);
Expanding base 'DC=avvid,DC=com'...
Result <0>: (null)
Matched DNs:
Getting 1 entries:
>> Dn: DC=avvid; DC=com;
            1> masteredBy: CN=NTDS Settings,CN=UNITY01,CN=Servers,CN=Default-
First-Site-Name, CN=Sites,CN=Configuration,DC=avvid,DC=com;
[…]
            1> fSMORoleOwner: CN=NTDS
Settings,CN=UNITY01,CN=Servers,CN=Default-First-Site-Name,
CN=Sites,CN=Configuration,DC=avvid,DC=com;
            1> gPLink: [LDAP://CN={31B2F340-016D-11D2-945F-
00C04FB984F9},CN=Policies,CN=System, DC=avvid,DC=com;0];
[…]
```

Attaques locales

Les attaques étudiées ici sont des attaques locales, au niveau du LAN principalement. Elles ont été testées sur des équipements Cisco IP Phone et leur infrastructure (équipements, serveurs et applicatifs).

Des vulnérabilités graves ont été mises en lumière et exploitées, allant d'un simple déni de service sur les équipements (téléphones et serveurs) au détournement de flux audio conduisant à l'écoute de n'importe quelle conversation sur le réseau, en passant par la récupération d'informations confidentielles et la facturation d'appels à d'autres personnes.

Utilisation frauduleuse des postes téléphoniques

Bien que la configuration locale des téléphones IP puisse être modifiée par l'utilisateur (en utilisant le code de déverrouillage '**#' par exemple sur les Cisco IP Phones), il est difficile d'effectuer une attaque à partir des postes téléphoniques IP. En effet, la plupart des paramètres ne sont pas modifiables localement. Seule une reconfiguration des interfaces du téléphone est possible.

Étanchéité des réseaux voix/données

L'IP Phone de Cisco, par exemple, intègre deux interfaces réseau et un switch interne, séparant le flux voix et le flux données dans deux VLANs différents. Dans l'utilisation normale d'un IP Phone Cisco, l'ordinateur de l'utilisateur est connecté sur la prise de données de son IP Phone, celui-ci étant lui-même connecté au réseau local de l'entreprise : figure 3 ci-dessous.

Figure 3

Tentatives de franchissement données vers voix

Dans la configuration ci-dessus, le franchissement données vers voix est impossible, car l'IP Phone procède à un filtrage du VLAN voix. Cependant, pour accéder au flux voix, il suffit de brancher un ordinateur directement sur la prise murale, avec des outils appropriés pour se placer sur le VLAN voix (par exemple noyau Linux avec extension protocole 802.1Q).

Tentatives de franchissement voix vers données

Le franchissement voix vers données n'a pas véritablement de sens dans la mesure où le flux de données n'est pas protégé par un VLAN spécifique. On accède donc aux mêmes informations, que l'on soit branché d'un côté ou de l'autre de l'IP Phone.

Bien sûr, le franchissement voix vers données n'est pas possible depuis le seul poste téléphonique.

Attaque des flux VoIP

Le flux voix VoIP utilise deux types de canaux de communication, l'un de contrôle, l'autre de données.

Le canal de contrôle utilise le protocole Cisco Skinny sur les ports TCP 2000-2002. Chaque téléphone IP établit une connexion TCP avec le serveur hébergeant le Call Manager 3.x sur le port 2000. Cette connexion est gardée active grâce à l'émission de « KeepAlive » toutes les 30 secondes.

Les caractéristiques importantes d'un point de vue sécurité sont les suivantes :

- → le canal de contrôle est connecté en permanence ;
- → la connexion est initialisée par le poste IP-phone ;
- → les actions Skinny sont transmises en clair.

Le canal de données sert à faire transiter les données audio des communications. Il utilise un dérivé du protocole RTSP : il s'agit de paquets UDP envoyés sur des ports dynamiquement négociés à travers le canal de contrôle.

Une communication classique comporte deux flux voix simultanés, ayant un débit de 50 paquets par seconde chacun. Chaque paquet est composé d'un en-tête de 54 octets (en-tête UDP + en-tête

RTSP) puis de 160 octets de données audio encodées en u-law 8bits, 8kHz.

Les caractéristiques importantes d'un point de vue sécurité sont les suivantes :

→ les ports UDP sont négociés dynamiquement au travers du canal de contrôle ;

→ les paquets sont numérotés séquentiellement dans l'en-tête RTSP ;

→ le flux audio est compressé mais non chiffré ;

→ l'absence de flux est autorisée et est synonyme de silence dans la communication téléphonique.

Insertion de paquets dans un flux VoIP

Si on est en mesure d'intercepter le flux d'un téléphone IP, il est possible d'insérer des paquets dans le canal de contrôle ou dans le canal de données, avec des résultats très différents.

Les attaques suivantes ont été réellement effectuées au cours de tests d'intrusion.

Insertion d'un paquet remplaçant un autre paquet de même taille dans le canal de contrôle

Lorsqu'il est possible de prédire l'émission d'un événement particulier dans le canal de contrôle par l'utilisateur d'un téléphone IP, il suffit d'envoyer, avec une très légère anticipation (quelques centièmes de seconde), un autre paquet qui sera pris en compte à la place du paquet prédit.

La substitution d'un paquet conduit aux résultats suivants selon les cas :

→ simuler l'appui d'une touche du pavé numérique du téléphone, pour modifier à la volée le numéro composé par l'utilisateur du téléphone (ou bien l'enregistrer afin d'espionner les numéros composés), ou d'effectuer des choix à sa place lors de la consultation d'un serveur vocal, par exemple ;

→ envoyer un faux paquet de type `OpenReceivedChannelAck`, juste avant l'émission du véritable paquet, contenant un mauvais numéro de port UDP pour spécifier le canal de donnée : l'utilisateur légitime n'entendra alors que du silence dans son combiné.

Insertion d'un paquet quelconque dans le canal de contrôle

Dans certains cas, on ne peut prédire avec certitude le prochain paquet émis sur le canal de contrôle par le Call Manager et reçu par le téléphone IP (ou vice versa).

Cependant, il est toujours possible d'envoyer un paquet qui sera pris en compte par le téléphone IP (un signal de sonnerie occupée par exemple). Mais dans ce cas, les numéros de séquence TCP étant erronés, le téléphone plantera quelques secondes plus tard, lors de l'émission du paquet suivant.

Déni de service généralisé

Nous avons vu qu'il suffit d'envoyer un paquet TCP de taille quelconque, mais doté d'un numéro de séquence correctement choisi, à un téléphone pour le rendre inutilisable jusqu'à son extinction manuelle.

Il est donc facile d'effectuer un déni de service généralisé sur l'ensemble du réseau VoIP.

Le scénario suivant pourrait par exemple être mis en œuvre :

1. écoute passive des *broadcasts* émis par les téléphones IP au moment de leur démarrage ;
2. identification d'un téléphone IP en train de booter ;
3. écoute passive de ce téléphone, à l'aide de ARP spoofing, le temps de récupérer la séquence TCP (30 secondes suffisent en général) ;
4. envoi d'un paquet TCP au téléphone IP, mettant celui-ci hors service.

Le réseau téléphonique entier serait alors hors service, pendant tout le temps que durerait l'attaque : à l'instant où un téléphone serait redémarré manuellement, l'attaque le mettrait à nouveau hors service de manière quasi immédiate.

La localisation de l'origine d'une attaque de ce type risquerait d'être longue et difficile.

Nous verrons les parades à ce type d'attaque au paragraphe sur les recommandations.

Détournement du trafic VoIP

En supposant que le LAN est commuté, il est nécessaire de recourir à la technique d'ARP spoofing pour intercepter le trafic associé à un téléphone IP. Cette technique consiste à envoyer sur le réseau Ethernet des paquets ARP forgés annonçant que l'adresse MAC possédant l'adresse IP cible est celle de notre carte réseau, et non celle de la machine cible réelle. Le résultat est que les paquets IP sont envoyés vers notre machine, et non plus vers la machine légitime. Nous pourrons alors « rerouter » le trafic reçu vers la machine cible, ou bien le faire disparaître.

Dans le cas de la VoIP, le but est :

→ de se faire passer pour le serveur Call Manager, la passerelle voix-données VG200 ou un autre téléphone IP distant vis-à-vis du téléphone IP à écouter ;

→ de se faire passer pour le téléphone IP à écouter vis-à-vis du Call Manager, de la passerelle VG200 ou du téléphone IP distant.

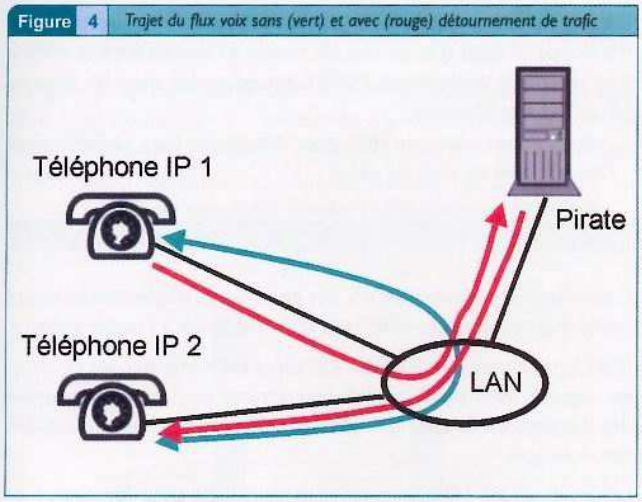
La figure 4 illustre ce mécanisme.

Dans cette configuration, tout flux reçu ou émis par le téléphone IP de la victime passe par l'ordinateur de l'attaquant. Celui-ci peut alors enregistrer la conversation, la modifier à la volée ou la renvoyer vers d'autres postes.

Il est tout à fait possible d'écouter un numéro de téléphone particulier. En fonction de ce numéro, on localise le téléphone IP correspondant en se connectant successivement à tous les téléphones pour accéder à son interface Web et récupérer le numéro. Une recherche complète sur un réseau de taille importante ne dure que quelques minutes.

Une fois le téléphone IP cible identifié (par son adresse IP), on se met en écoute et on attend l'établissement d'une communication. À ce moment, l'ARP spoofing est lancé afin d'intercepter le flux voix. Ce flux étant récupéré de manière transparente, le processus est indétectable par les utilisateurs espionnés. L'assemblage des

Figure 4 *Trajet du flux voix sans (vert) et avec (rouge) détournement de trafic*



paquets audio ne pose en général pas de problème pour reconstituer la conversation d'origine. Bien souvent, il suffit d'additionner le signal circulant dans un sens et dans l'autre pour reconstituer la « bande son » de la conversation entre les deux protagonistes, qu'ils soient tous deux situés dans l'entreprise ou que l'un d'entre eux seulement s'y trouve.

L'écoute téléphonique est donc possible à large échelle sur un réseau local VoIP. Il suffit de se connecter à une prise réseau pour écouter l'intégralité des communications émises ou reçues.

Attaques « génériques »

Anonymat

Dans la majorité des échanges informatiques, les différentes adresses des correspondants (Mails, IPs, FQDN, etc.) ne sont pas des facteurs déterminants, au contraire du contenu des transactions.

Pour la VoIP, les adresses « directes » sont au premier plan, au même titre que les numéros de téléphone RTC, d'où l'apparition de la fonction « numéro caché » et de la liste rouge !

L'anonymat des correspondants n'est pas forcément simple à mettre en œuvre, car il devient difficile de filtrer le trafic et de cohabiter avec des règles strictes de filtrage.

Les relais SIP permettent d'assurer l'anonymat des utilisateurs, dans une certaine mesure (car limités à des plages d'adresses la plupart du temps), en se référant au *Call-ID* qui doit être présent dans tous les paquets de la même transaction pour qu'ils soient recevables. En revanche, les dénis de service (par envoi continu de SIP INVITE par exemple) sont alors plus difficiles à contrer.

L'anonymat ne peut donc pas être assuré dans le cas d'une communication VoIP directe sans *proxy* entre 2 clients.

Spoofing

Dans un paquet SIP, les identifiants d'une communication, lorsqu'elle est établie, sont le *Call-ID* et les *tags* des champs *From* et *To* s'ils sont utilisés. Les 3 principaux types de *spoofing* sur le protocole SIP s'effectuent avec des paquets SIP INVITE, BYE et CANCEL (cf. figure 1).

Le premier est trivial à mettre en œuvre en LAN comme sur Internet en forgeant un paquet SIP INVITE (avec des champs *Call-ID*, *From*, *To*, et *Cseq* adéquats). L'hôte distant sonnera, mais la conversation ne pourra s'établir que si le *tag* du champ *To* est identique à celui retourné par l'appelé lorsqu'il accepte l'appel. Malheureusement, de nombreuses implémentations de SIP n'utilisent pas les *tags* et sont donc vulnérables à ces attaques, même lorsque l'attaquant n'est pas en mesure de renifler le trafic.

Dans le paquet SIP, le champ *From* est suivi d'un *tag*. Lors de la réception du SIP INVITE, l'appelé ajoutera également son *tag* aléatoire et toutes les communications suivantes devront inclure ces 2 *tags* respectifs sur *From* et *To*.

La probabilité de réussite pour falsifier les deux paquets suivants est quasiment nulle sans écoute du trafic si l'aléa du *Call-ID* et des *tags* sont corrects.

Dans le cas d'un LAN où il est possible d'écouter le trafic réseau, et donc de récupérer les *Call-IDs*, *From*, *To*, et les *tags* s'ils sont utilisés, tous les types d'attaques sont envisageables, du MITM au DoS, en passant par une écoute passive temps réel des conversations ou une modification à la volée des paquets. Seule la voix pourrait alors faire défaut... !

Et sur Internet ?

Comme nous l'avons vu, bon nombre d'attaques se font grâce à de l'ARP spoofing ou du DNS spoofing. L'attaque ARP n'est pas très réaliste sur l'Internet et c'est pourquoi l'interception de la communication se fera plutôt en utilisant d'autres moyens.

La proximité (au sens réseau) du pirate et de la source ou de la destination est un facteur clé pour faciliter l'attaque et l'écoute.

L'attaque la plus commune reste le déni de service (mutualisé) à l'encontre du lien d'accès à l'Internet ou de la passerelle VoIP-PSTN/LAN. Toujours aussi simple à mettre en œuvre et complexe à filtrer et à tracer.

En ce moment, de petits malins recherchent activement ce genre de plates-formes VoIP-PSTN connectées à l'Internet et mal configurées : cela leur permet de terminer gratuitement leur communication VoIP via l'Internet sur le réseau téléphonique commuté dans différents pays...

« Ajouts » et recommandations sécurité

RTP (*Real Time Transport Protocol*) encode, transporte et décode la voix. La qualité du flux étant essentielle dans le domaine de la VoIP, tant sur le plan de la vitesse que sur la qualité, le compromis sécurité/utilisation est donc essentiel : débit, qualité de la voix, temps d'établissement des communications, etc.

Certaines solutions classiques ne sont donc pas viables et le meilleur compromis se détermine au cas par cas selon le nombre de clients, les débits souhaités, le niveau de sécurité requis, la vitesse du média utilisé, les types de données...

Les contraintes d'intégrité, de confidentialité et d'authenticité ne tenaient pas une place de choix dans les premières solutions et les protocoles clefs ne disposaient d'aucune protection fiable (SIP, RTP, RTCP, etc.). Les problèmes s'accentuaient avec l'utilisation massive

d'UDP pour accélérer les échanges, entre autres avec les problèmes évidents de *spoofing*.

La VoIP permet de remplacer la totalité des lignes RTC classiques (avec l'utilisation de PABX-IP), et c'est alors que la sécurité de l'architecture est rentrée dans le cahier des charges.

Plusieurs protocoles sont apparus avec notamment les équivalents chiffrés de RTP et RTCP : SRTP et SRTCP [1] (respectivement *Secure Real-Time Transport Protocol* et *Secure Real-Time Transport Control Protocol*). Les paquets SRTP se différencient des paquets RTP par 3 champs :

- → un champ additionnel pour le type d'algorithme utilisé : *Authentification tag* ;
- → un deuxième contenant différentes informations sur la clef : *Master Key Identifier (MKI)* ;
- → et bien sûr un *payload* chiffré.

SRTP et SRTCP ont été développés dans un souci de performances, le but étant de sécuriser au maximum les échanges à moindre coût en minimisant la surcharge liée au chiffrement du *payload*.

Paquet SRTP

...
Synchronization Source (SSRC) identifier
Contributing Source (CSRC) identifiers
RTP extension
PAYLOAD (CHIFFRÉ)
MKI
Authentification tag

Comme le montre la figure ci-dessus, seul le *payload* est chiffré dans un paquet SRTP, ce qui ne permet donc pas d'assurer à 100% l'intégrité des paquets transmis : l'en-tête du paquet SRTP pourrait être modifié, voire les champs optionnels *MKI* ou *Authentification tag*.

Quand SRTP est utilisé conjointement avec SIP [2], le déroulement chronologique des transactions est le suivant :

Appelant (*ringing*)	établissement de la communication avec un paquet SIP INVITE.
Appelé	accord du tiers distant avec une réponse 200 (OK).
Appelant	paquet SIP de type ACK pour confirmer la réception du paquet précédent et établir la session SIP.

Le schéma d'établissement ressemble étrangement à une ouverture de session TCP... Une fois la communication établie, et si les deux participants se sont préalablement mis d'accord sur l'algorithme de chiffrement utilisé, alors la communication sécurisée est établie.

Dans le cas contraire, si une négociation des clefs est nécessaire, un protocole de gestion des clefs s'impose sur le même principe, par exemple, qu'IKE pour IPSec.

C'est dans ce but que MiKEY [3] (*Multimédia Internet Keyring*) a été développé : il s'agit d'un protocole récent à l'état de *draft* et encore très rarement implémenté. MiKEY est encapsulé dans les paquets SIP et permet d'utiliser :

- → un secret commun (*PSK* pour *Pre-Shared Key*), généralement sous forme de mot de passe ;
- → des protocoles Diffie-Hellman d'échange de clés ;
- → une PKI.

Cette dernière alternative n'a pas encore été implémentée et ses performances globales sont très controversées à l'heure actuelle.

MiKEY, tout comme SRTP/SRTCP, tente de minimiser les coûts et les impacts de la protection. Il doit assurer une sécurité optimale des transactions de clefs sans affecter de façon significative la rapidité des échanges.

Le projet *minisip* [4], bien qu'encore trop jeune pour être utilisé en production, est l'un des plus avancés dans ce domaine à l'heure actuelle. Il s'agit d'un client implémentant MiKEY et SRTP avec au choix une authentification *PSK* ou Diffie-Hellman. Il supporte également l'utilisation de TLS pour sécuriser les échanges SIP.

MiKEY s'encapsule dans SIP avec le champ `a=key-mgmy` qui permet d'assurer l'authentification qui permettra de faire transiter un flux SRTP par la suite avec des algorithmes et des clefs adéquats :

Paquet SIP INVITE + MiKEY

```
INVITE sip:lolo@rstack.org;user=phone SIP/2.0
From: <sip:mp@domain.org;user=phone>;tag=1017556314
To: <sip:lolo@rstack.org;user=phone>
Call-ID: 1664131130@rstack.org
CSeq: 101 INVITE
Contact: <sip:mp@rstack.org:5060;user=phone;transport=UDP>;expires=900
Content-Type: application/sdp
Via: SIP/2.0/UDP 192.168.0.1:5060
[...]
m=audio 32945 RTP/AVP 0 97
a=key-mgmt:mikey  AQAFgAAAAAsAxNbayXB+WngBEH (...) jPANqgLOmmfPv0+/f56vA==
a=rtpmap:0 PCMU/8000/1
a=rtpmap:97 iLBC
```

En utilisant des relais SIP, lorsque **mp@rstack.org** cherche à contacter **lolo@rstack.org**, le relais sortant effectue une requête DNS de type SRV pour connaître l'IP du serveur SIP distant, par exemple celle de **sip.domaine.org**.

Cette requête est une cible idéale pour un pirate, par exemple via une attaque de *spoofing* sur le *DNS ID* ou de *cache poisoning* [5] afin de renvoyer l'adresse d'un *proxy* SIP qu'il contrôle. L'utilisation de DNSSEC pourrait donc être envisagée à ce niveau.

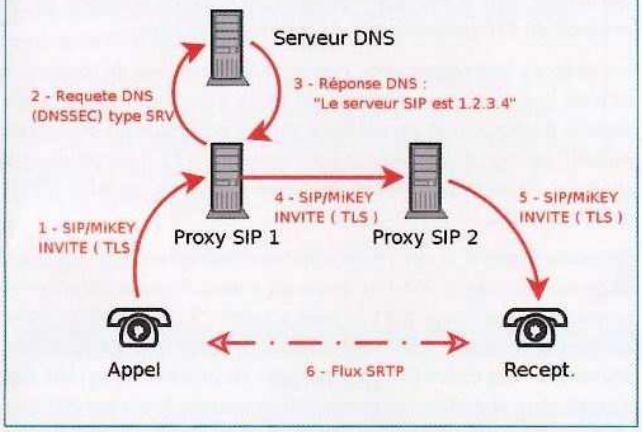
Comme nous l'avons vu, MiKEY repose sur SIP : toute attaque sur ce dernier remet donc en cause la sécurité.

Il est indispensable de veiller à l'intégrité et l'authenticité des paquets SIP. La nécessité pour certains intermédiaires d'accéder, voire de modifier des portions de paquets (*proxies*, *registrars*, *redirecteurs*, *etc.*)

rend la tâche délicate. *Minisip* propose l'utilisation de TLS pour limiter ces risques.

La figure 5 représente une solution sécurisée d'architecture de VoIP, mais il en existe bien sûr beaucoup d'autres. *Minisip* permet de tester cette solution dans son intégralité. Pour plus de détails, vous pouvez consulter [6].

Figure 5 *Solution sécurisée, également celle retenue par le projet minisip...*



La VoIP faisant appel à de nombreux processus (SIP, DNS, SRTP, voire SRTCP pour le contrôle du flux SRTP : CODECs, *timing*, etc.), il est indispensable de sécuriser chaque étape de la communication.

Les nombreuses contraintes imposées par cette technologie ne facilitent pas la tâche : il est aussi parfois nécessaire de traverser des pare-feu, de fonctionner avec des translations d'adresses (NAT), et certains choix sont alors limités.

Tunnel IPsec

Les différentes solutions de *tunneling* présentent toutes des avantages et des inconvénients pour la VoIP avec ses nombreuses exigences. IPSec, qui travaille sur la couche réseau, permet d'assurer une plus grande fiabilité des informations. Notons par exemple que le problème des en-têtes SRTP modifiables n'est plus un souci ici.

Cependant, le coût de cette solution est parfois considérable, tant sur le plan des ressources matérielles que sur le trafic réseau. IKE (*Internet Key Exchange*) permet alors de remplacer MiKEY et d'assurer la gestion des clefs pour l'ensemble des communications VoIP.

La surcharge engendrée par IPSec peut être minimisée en configurant le tunnel pour traiter uniquement les flux de voix sur IP (pour des machines/protocoles fixés). Un atout intéressant est la possibilité d'utiliser la totalité des *soft phones* disponibles puisqu'ils n'ont plus à gérer la sécurité des échanges (via SRTP/MiKEY...).

UDP limite les types de tunnels utilisables, notamment SSL ou SSH, même s'il reste possible d'utiliser `vtun` (ou un équivalent) pour faire de l'*UDP over TCP*, mais les performances deviendraient rapidement médiocres !

Pour résumer : les tunnels simplifient le déploiement de la VoIP sécurisée, mais ne peuvent pas être employés sur de larges infrastructures ou sur des *soft phones* peu puissants.

Filtrage réseau

Comme nous l'avons vu plus haut, les serveurs de gestion VoIP (surtout installés par défaut) ont un nombre de ports ouverts par défaut très conséquent.

Il est donc fortement recommandé de filtrer les ports accessibles sur les serveurs depuis le réseau des utilisateurs, au niveau des routeurs. Pour cela, il peut être utile de placer les serveurs sur un sous-réseau dédié.

Il convient de lister les services et les ports associés qui doivent être pris en considération lors de l'implémentation d'une politique de filtrage sur un réseau VoIP. Certains protocoles sont propriétaires (ex : Cisco Skinny) et d'autres ne font pas bon ménage avec du filtrage sans état (non-*stateful*) ou si des relais applicatifs, qui savent décoder le protocole, ne sont pas présents (ALGs – *Application Level Gateways*). Attention, bon nombre de pare-feux se limitent à gérer l'ouverture de ports en fonction des communications et n'inspectent pas les flux (au niveau protocolaire – i.e. AGLs). De plus, cet élément additionnel risque d'introduire un délai ainsi qu'une gigue, c'est pourquoi ils sont absents dans bien des déploiements.

IDS

Il n'est pas très courant de trouver des outils de détection d'intrusion pour des solutions de voix sur IP. La quantité de faux positifs dus à l'observation du flux RTP pourrait être plus que conséquente. Bien qu'elle permette de détecter des dénis de service par exemple, la détection d'intrusion se ramène souvent à de la détection de fraude.

Recommandations

Les principales recommandations permettant de se protéger contre les attaques précédentes sont les suivantes :

- → verrouiller les adresses MAC/adresses IP par port dans les commutateurs traversés par du flux VoIP (cela peut être relativement simple lorsque les téléphones IP ne se déplacent jamais) ;
- → utiliser IPSec entre les équipements VoIP ;
- → mettre à jour les applicatifs et les *firmwares* des équipements VoIP ;
- → activer le chiffrement des communications VoIP ;
- → utiliser l'authentification des équipements VoIP ;
- → activer au mieux les fonctions de sécurité offertes par les équipements de l'infrastructure VoIP utilisée et qui ne seraient pas activées par défaut.

En ce qui concerne l'architecture Cisco, la nouvelle version du Call Manager (4.x) introduit des améliorations notables en matière de sécurité :

- → signature du code des *firmwares* et vérification lors du téléchargement sur les IP Phones ;
- → authentification réciproque des IP Phones et des serveurs Call Manager par certificats (un certificat unique est installé d'origine dans chaque IP Phone, et une *Certificate Trust List* ou CTL est utilisée) ;
- → authentification de la signalisation (TLS est utilisé afin de vérifier que les paquets de signalisation n'ont pas été modifiés) ;

→ chiffrement des communications (à l'aide du protocole SRTP, qui utilise des clés asymétriques pour chiffrer et authentifier les paquets de données audio) ;

→ possibilité de désactiver certaines fonctionnalités des IP Phones (désactivation du port de connexion au PC, désactivation du *broadcast* de l'adresse ARP du téléphone au démarrage, désactivation de l'accès au VLAN voix, désactivation de l'accès à certaines pages de configuration et de statistiques).

Il est donc fortement recommandé de migrer une architecture CCM 3.x en version 4.x.

Interception « légale » de trafic

L'interception légale de trafic est une fonction/interface qui est présente dans la majorité des réseaux téléphoniques (fixes et cellulaires). Pour l'instant, cette obligation légale est encore au stade de discussions dans beaucoup de pays et les autorités de régulations ont des idées assez divergentes : voir [18] et [19].

Une anecdote assez intéressante : un nombre conséquent de personnes pensent que les agences de renseignements doivent avoir de plus en plus de mal à intercepter les communications vu le nombre de médias et la quantité d'informations.

Un commentaire d'un haut responsable d'une telle agence suggère tout le contraire : en effet, « avant », il fallait numériser toutes ces informations pour devoir les traiter, aujourd'hui cette étape conséquente se fait aux extrémités, ce qui leur simplifie largement la tâche…

Les réseaux de téléphonie « classiques »

A titre de comparaison, nous allons discuter, sans entrer dans les détails, de la sécurité des réseaux téléphoniques plus classiques.

PSTN/POTS

Le réseau téléphonique filiaire (PSTN/POTS – *Public Switched Telephone Network/Plain Old Telephone System*) transporte voix et données. A la différence d'un réseau VoIP, où les équipements qui « gèrent » les communications sont souvent dans le même réseau et accessibles, cela n'est pas le cas dans le RTC (SS7 et le « switch » par exemple).

Il est bien connu que la majorité des téléphones sans fil analogiques peuvent être écoutés, mais que la distance est limitée à quelques centaines de mètres. Les téléphones sans fil du type DECT peuvent chiffrer la communication.

GSM

Le réseau cellulaire (GSM – *Global System for Mobile Communications*), à la différence d'un réseau sans fil de type 802.11, ne permet pas d'écouter facilement une communication. En revanche, les communications ne sont chiffrées qu'entre le téléphone de l'utilisateur et la station à laquelle il est rattaché.

Ce n'est pas un chiffrement de bout en bout entre l'appelant et l'appelé, mais des solutions existent, qui emploient par exemple le mode données pour transporter de la voix chiffrée (voir [21]).

Un changement majeur est apparu ces dernières années avec le déploiement des réseaux de troisième génération (GPRS et UTMS) : les téléphones ne sont plus des grille-pains, mais sont livrés avec un système d'exploitation, Java, une pile TCP/IP, etc. De plus, ils sont connectés en permanence au réseau et disposent d'une adresse IP.

Skype

Skype [7] est un logiciel récent développé par l'équipe de *KaZaA* qui innove dans la VoIP en proposant un système reposant sur le principe du *P2P* (*peer-to-peer*) via le réseau *FastTrack*.

Les clients s'interrogent pour connaître les « réseaux de contacts » et ainsi communiquer directement entre eux. *Skype* ne demande aucune configuration particulière sur les équipements du réseau puisqu'il nécessite uniquement des connections TCP sortantes vers les ports 80 ou 443, par exemple. Il gère également les relais HTTP classiques.

En mode *PC-to-PC*, la communication est chiffrée en AES (*Advanced Encryption Standard*) 256 bits entre les clients. Reste à déterminer le niveau de confiance que l'on peut accorder à un algorithme dont on ne maîtrise pas l'implémentation : le code source n'est pas disponible ! Les clefs AES sont négociées en utilisant RSA (1536 bits à 2048 bits). *SkypeOut*, qui permet de contacter le réseau RTC, ne chiffre pas les échanges : il aurait pu être possible cependant de le faire jusqu'au PABX-IP distant…

Skype semble offrir une belle perspective à la VoIP sur Internet pour un usage personnel. Les contraintes de sécurité ont été prises en compte, ce qui est souvent rare dans ce type de projets. En revanche, « l'opacité » de *Skype* sur les protocoles qu'il utilise ou sur son fonctionnement le rend difficile à utiliser dans un contexte professionnel, surtout si la confidentialité est au centre des préoccupations.

D'autres projets, comme SIPshare [18], s'intéressent par exemple à l'utilisation de SIP dans un réseau P2P.

VoWLAN

VoWLAN, comme son nom l'indique, n'est rien d'autre que de la VoIP appliquée sur des réseaux sans fil. Elle offre encore de nouvelles perspectives, notamment avec l'utilisation de PDAs ou de téléphones SIP compatibles WLAN. La qualité peut alors largement dépasser celle des téléphones portables cellulaires et il est possible de couvrir de grandes distances avec plusieurs AP en *roaming*. Le protocole IAPP (*Inter Access Point Protocol*) assure la normalisation de cette technologie pour passer d'AP en AP de façon transparente.

En revanche, l'utilisation du WiFi pose certains problèmes, notamment sur les questions de débit et de sécurité... Des CODECs appropriés sont nécessaires (PCM, GSM, etc.) pour assurer une qualité minimale sur une bande passante réduite, en fonction de différents critères, dont la compression, les délais (établissement, latence, etc.), et la qualité (écoute, écho, pertes, etc.).

La sécurité du réseau WiFi doit être assurée au préalable (cf. [8]) tout en garantissant un débit minimum afin d'éviter les surcharges liées à la sécurisation du réseau sans fil (WEP/WPA, IPSec, tunnels, etc.).

La VoWLAN présente tous les enjeux de sécurité de la VoIP auxquels viennent s'ajouter ceux des réseaux sans fil.

Conclusion

Nous avons couvert le sujet de la voix sur IP d'un point de vue technique et nous pensons qu'aujourd'hui, une solution de voix sur IP peut être sécurisée à un niveau acceptable.

Un projet de voix sur IP est complexe, car il n'existe pas de solution générique, et une étude au cas par cas s'impose avant la mise en œuvre de cette technologie. Le facteur sécurité doit être pris en compte avant même la phase de conception, en posant les bonnes questions aux vendeurs que vous êtes en train de sélectionner.

La convergence des deux réseaux (informatique et téléphonie) change la donne pour la partie voix. Il convient de définir clairement quel département est responsable de la sécurité des parties et de l'ensemble, ce qui bien trop souvent n'est pas fait.

Références

[1] RFC 3711 : SRTP et SRTCP

[2] RFC 3261 : SIP

[3] RFC (*draft*) MiKEY - draft-ietf-msec-mikey-08.txt

[4] minisip- *Soft phone* SIP/MiKEY - **http://www.minisip.org**

[5] *« Failles intrinsèques du protocole DNS »* **http://betouin.security-labs.org/Article_DNS_PBT.pdf**

[6] *« Secure Mobile Voice over IP »* - **ftp://ftp.it.kth.se/Reports/DEGREE-PROJECT-REPORTS/030626-Israel_Abad_Caballero-final-report.pdf**

[7] Skype - **http://www.skype.com**

[8] "La sécurité des réseaux 802.11 : quoi de neuf depuis un an ?" - MISC n°12

[9] Sécurisation de Windows NT/2000/XP/2003 - **http://www.chambet.com/publications/sec-win2k/**

[10] Sécurisation d'IIS - **http://www.chambet.com/publications/iis-security/**

[11] Vulnérabilités et sécurisation des applications Web - **http://www.chambet.com/publications/sec-web-apps/**

[12] RFC 1189 : RTP

[13] H.235 : **http://www.javvin.com/protocolH235.html**

[14] H.323 : **http://www.h323forum.org/papers/**

[15] Sécurisation des routeurs et des commutateurs Cisco - **http://www.miscmag.com/articles/index.php3?page=214**

[16] PROTOS SIP - **http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/sip/**

[17] **http://www.voip-info.org/wiki-Asterisk+security**

[18] CALEA - **http://www.fcc.gov/calea/**

[19] ETSI and LI - **http://www.ripe.net/ripe/meetings/ripe-48/presentations/ripe48-eof-etsi.pdf**

[20] SIPshare - **http://www.research.earthlink.net/p2p/**

[21] GSMK Cryptophone - **http://www.cryptophone.de/**

Sécurité des téléphones portables

Introduction

Suite aux nombreuses annonces faites ces derniers temps autour du code malveillant pour équipements mobiles, et dans un contexte d'incertitude générale sur la faisabilité réelle d'une infection à grande échelle, cet article vise à poser les bases du développement sur téléphones portables, les capacités des équipements actuels, mais aussi leurs limites.

Le sujet de la sécurité des téléphones portables n'est pas nouveau, puisqu'en l'an 2000 déjà, le virus VBS/Timofonica [1] avait pour charge active d'envoyer des SMS aux abonnés du service Movistar. Ce virus se propageait toutefois exclusivement sur PC et n'avait pas d'impact sur les téléphones visés.

Le sujet connaît un regain d'intérêt depuis l'année dernière, avec la généralisation des plates-formes Symbian OS et Windows CE, beaucoup plus performantes en termes de programmation que les anciens téléphones « propriétaires ». Des codes malveillants destinés aux équipements mobiles sont apparus cette année, et le sujet a été abordé lors de Black Hat Europe 2004, preuve de l'activité des chercheurs dans le domaine.

Cet article s'organise en trois parties ; tout d'abord les motivations des potentiels attaquants seront passées en revue ; les différents environnements de développement (assembleur, Java, C++) accessibles sur téléphone portable seront ensuite présentés ; enfin des scénarios d'attaque crédibles seront construits sur la base de ces éléments techniques.

Cet article se conclura sur une présentation des principaux codes malveillants existant actuellement, tels que les vers Cabir et Dust, le cheval de Troie Brador, l'application Mosquito et les attaques BlueTooth. Pour une analyse plus technique de ces menaces, se reporter à l'article d'Eric Filiol dans ce même numéro.

A titre d'information, et sans vouloir faire de publicité aucune, les tests présentés dans cet article ont été réalisés sur téléphones Sony Ericsson T610, Nokia N-Gage et Sony Ericsson P900, c'est-à-dire des plates-formes Symbian OS.

Attaquer les téléphones portables, pour quoi faire ?

Avant d'entrer dans toute considération technique, il convient de s'interroger sur les objectifs que poursuivrait raisonnablement un « chercheur » en sécurité des téléphones portables, afin de bien cerner les menaces réelles des menaces fantasmées.

Pouvoir faire décrocher silencieusement un téléphone à distance afin d'écouter les conversations environnantes semble être la menace la plus effrayante. Des téléphones modifiés offrent déjà une telle fonctionnalité [2], mais elle n'est pas la plus « rentable » pour un attaquant de masse, loin s'en faut ...

Il existe *a priori* deux moteurs qui motivent les attaquants : la gloire et l'argent. Passons en revue, dans les 2 tableaux ci-contre, quelques scénarios (cette liste n'est pas limitative) qui permettraient d'atteindre l'un ou l'autre, en utilisant du code malveillant ou simplement des techniques d'ingénierie sociale, très efficaces dans le monde « magique » de la téléphonie mobile ! (Qui n'a jamais entendu parler du fameux correspondant qui vous appelle en vous demandant de taper un code sur le clavier de votre téléphone ? [3])

Comme on le voit dans le tableau ci-contre, toutes les attaques ne sont pas techniques, loin s'en faut. Un de mes exemples préférés est le suivant :

1. Appeler un ensemble d'abonnés depuis un numéro surtaxé ;
2. Laisser sonner une fois et raccrocher ;
3. Un nombre non négligeable d'utilisateurs, voyant un appel en absence, vont avoir la curiosité de rappeler le numéro affiché : ils se retrouvent alors en communication avec un numéro surtaxé.

Dans la suite de l'article, on s'intéressera exclusivement aux moyens reposant sur une attaque technique du téléphone portable. Ceci exclut toute forme d'ingénierie sociale, d'attaque du réseau des opérateurs, de scan des passerelles GSM et/ou SMS, etc.

Qu'est-ce qu'un téléphone portable ?

Matériel

Au niveau matériel, la plupart des téléphones portables actuels sont architecturés autour d'un processeur RISC de la famille ARM [4]. Par exemple, la nouvelle console N-Gage QD intègre un processeur ARM9 à 104 MHz, ce qui lui confère environ 120 MIPS de puissance de calcul.

A l'instar de la plate-forme PowerPC, la plate-forme ARM est relativement ouverte et de nombreux fondeurs proposent des processeurs compatibles intégrant des fonctions « on-chip » supplémentaires, telles que DSP, support PCMCIA, etc. On peut citer par exemple les processeurs SA-1110 et son successeur le PXA255 chez Intel, le OMAP 730 chez TI qui équipe certains SPV, la série i.Mx chez Motorola, etc. [5]

Système d'exploitation

Au niveau logiciel, les modèles les plus anciens (ce qui représente environ 2 ans en matière de téléphonie mobile) ou les moins chers intègrent encore des systèmes d'exploitation propriétaires.

Nicolas Ruff
nicolas.ruff@edelweb.fr
Consultant en Sécurité Windows et Codes Malveillants

Gloire

#	Objectif	Exemples de moyens
1	Ecrire dans MISC	■ Tous les moyens sont bons, cet article en est la preuve ;-)
2	Provoquer un déni de service de suffisamment grande envergure pour être cité dans les médias.	■ Créer un code malveillant* capable de : ▪ bloquer un grand nombre de téléphones au redémarrage, ▪ envoyer massivement des SMS, ▪ provoquer un comportement aléatoire du téléphone (envoi de numéros du carnet d'adresses à des adresses aléatoires), ▪ vider rapidement la batterie du téléphone en effectuant des opérations gourmandes en énergie, ▪ etc.

* Dans ce contexte, la notion de code malveillant est à prendre au sens large : il peut s'agir d'un virus, mais aussi d'une application installée volontairement par l'utilisateur tel qu'un jeu. On retrouve la notion de « malware » de plus en plus présente dans l'informatique « traditionnelle », qui regroupe vers, virus, spywares et autres saletés.

Argent

#	Objectif	Exemples de moyens
3	Récupérer des informations à valeur commerciale sur le téléphone de l'abonné (essentiellement son carnet d'adresses, qui peut être utilisé pour du spam SMS ou de la constitution d'annuaire).	■ Créer un code malveillant. ■ Inciter l'utilisateur à saisir des numéros de téléphone sur un site Web en lui faisant miroiter des services gratuits ou un système de parrainage.
4	Récupérer des informations confidentielles opérateur, tel que l'IMEI de l'abonné ou son code PIN.	■ Créer un code malveillant dialoguant avec la carte SIM ou simulant un démarrage du téléphone.
5	Se connecter à des services surtaxés (numéros de téléphone ou SMS).	■ Créer un code malveillant émettant des SMS ou des appels vers des numéros surtaxés. ■ Inciter l'utilisateur à se connecter lui-même à de tels numéros, par le biais de techniques d'ingénierie sociale (cf. ci-après).
6	Utiliser le téléphone de l'abonné en rebond pour se connecter à des services payants (de nombreux services, tels que les accès WiFi, sont déblocables par envoi de SMS et facturés directement à l'abonné).	■ Créer un code malveillant ayant la capacité de recevoir des commandes via GPRS et de le retransmettre par SMS. ■ Exploiter une faille BlueTooth pour faire émettre des SMS à un téléphone proche.
7	Insérer de la publicité sur le téléphone de l'abonné (principe existant déjà dans de nombreux « spywares » sur PC).	■ Créer un code malveillant, qui une fois installé, télécharge par GPRS des bandeaux publicitaires affichés en fond d'écran.
8	Rétro-ingénierie des fonctions téléphoniques.	■ Inverser le code installé dans un téléphone pour en copier les fonctions (ex. algorithmes de reconnaissance vocale ou d'écriture intuitive).
9	Espionnage industriel.	■ Créer un code malveillant capable de : ▪ récupérer à distance les derniers numéros appelés, ▪ enregistrer les appels vocaux dans la mémoire interne du téléphone et les retransmettre par GPRS, ▪ activer à distance les fonctions photo/vidéo/son du téléphone.

Toutefois, la majorité des téléphones modernes intègrent un système d'exploitation qui est dans la majorité des cas Symbian OS ou Windows CE.

Symbian OS

Ce système a été initialement conçu pour les machines Psion, et renommé Symbian OS à partir de la v5.

Il existe actuellement 3 versions majeures en circulation : la v5, la v6 (la plus courante, qui équipe la majorité des Nokia) et la v7. À l'heure où j'écris ces lignes, la v8 vient de sortir.

Windows CE / Windows Mobile / Windows Embedded

Comme son nom ne le suggère pas, ce système embarqué n'a pas grand-chose à voir avec le Windows XP qui équipe la plupart des postes de travail, ni même avec Windows XP Embedded.

Il s'agit d'un système d'exploitation léger et modulaire, supportant différents processeurs dont les processeurs ARM, et dont le code source est disponible sur : **http://msdn.microsoft.com/embedded/** !

Toutefois, les API disponibles restent assez proches des API Windows traditionnelles, ce qui facilite le portage d'applications.

Actuellement, il existe 3 versions majeures en circulation : la v3, la v4 (dite .NET) et la toute nouvelle v5. Ce système a bien d'autres noms, tels que Windows Mobile 2003 ou Pocket PC 2003, mais tous ces OS reposent sur la même base de code.

Développement d'applications pour téléphones portables

Assembleur ARM

Sans rentrer dans les détails de l'assembleur ARM, de très bons tutoriaux étant disponibles sur Internet [6], voici néanmoins quelques clés pour comprendre vos listings IDA Pro :

- Les instructions sont de taille fixe.
  - 16 bits en mode THUMB et 32 bits en mode ARM, le mode de fonctionnement étant modifiable à tout moment via le « state bit » du registre CPSR.

- Toutes les instructions sont conditionnelles, ce qui permet des optimisations assez déroutantes pour un être humain.
  - L'exécution de n'importe quelle instruction (ADD, SUB, MUL, etc.) est conditionnée par l'état des drapeaux du processeur (Zero, Carry, Overflow, etc.). La forme « courante » de l'instruction n'est qu'un alias sur la condition « Always » (ADD = ADDAL, SUB = SUBAL, etc.).

- Le processeur possède 7 états de fonctionnement – correspondant aux Ring 0 à 3 sur les processeurs Intel – et accessibles via le registre CPSR également.
  - Ce sont les modes User, FIQ, IRQ, Supervisor, Abort, System et Undefined.

- 16 registres sont utilisables simultanément (R0 à R15), un système de masquage de registre permettant de préserver la valeur des registres clés entre les différents modes processeur.
  - R15 est utilisé comme pointeur d'instructions (PC, *Program Counter*)
  - R14 comme adresse de retour (LR, *Link Register*)
  - R13 comme pointeur de pile (SP, *Stack Pointer*)

- L'instruction d'appel système est l'instruction SWI <*n*>. Le numéro de l'appel système n'est pas passé dans un registre, comme sur plate-forme Intel.
  - La méthode « officielle » pour obtenir le numéro de l'appel système en cours est de lire l'opcode de l'instruction SWI qui a provoqué l'appel, à l'emplacement LR-4.

Comme sur toute plate-forme RISC, le compilateur C effectue un gros travail d'optimisation, et le code généré est parfois difficile à comprendre par un œil non habitué. On pense par exemple au chargement de constantes (il est impossible de charger une valeur 32 bits dans un registre en une seule opération) ou à la construction des tables de « switch ».

Java

La quasi-totalité des téléphones portables, y compris les modèles les plus anciens (sans système d'exploitation ouvert), embarquent une machine virtuelle Java : au minimum la KVM (*Kilobyte Virtual Machine*). Comme son nom l'indique, il s'agit d'une machine extrêmement allégée, dont les spécifications sont disponibles sur le site de Sun, et capable de fonctionner avec uniquement 128 Ko de RAM [7].

Les téléphones les plus « gros » peuvent embarquer des machines plus « complètes », c'est-à-dire mettant à disposition du développeur une API bien plus riche. A ce sujet, il faut préciser que si l'API de base a été standardisée, de nombreuses extensions ont été proposées par les constructeurs pour supporter des fonctions exotiques telles que clapets, vibreurs, molettes, et autres. Ceci explique pourquoi les applications (et en particulier les jeux) Java les plus élaborées sont spécifiques à une famille de téléphones donnée, malgré l'apparente universalité de ce langage.

Les 4 niveaux d'API Java potentiellement disponibles sur les téléphones sont à l'heure actuelle :

1. CLDC (*Connection Limited Devices*) v1.0 et v1.1 :

la base minimale des API disponibles.

2. MIDP (*Mobile Information Device Profile*) v1.0 et V2.0 :

une nouvelle base minimale d'API, sur-ensemble du précédent. Ce standard est aujourd'hui largement implémenté dans les téléphones du commerce et on trouve peu d'équipements compatibles CLDC uniquement. Les applications compatibles MIDP sont appelées « MIDlets ».

3. JSR (*Java Specification Requests*) :

des extensions d'API normalisées ou en cours de normalisation. Celles-ci sont disponibles sur le site **http://www.jcp.org/**. A titre d'exemple, on peut citer :

→ Les chaînes de caractères sont de type Pascal par défaut (préfixées avec leur taille).

→ Etc.

La création de shellcode Linux/ARM ayant été traitée dans Phrack n°58, j'invite le lecteur curieux à s'y reporter. La création de shellcode Symbian OS n'a fait l'objet d'aucune étude à ma connaissance.

❑ *Exemple de code ARM vu avec IDA Pro 4.7*

```
.text:1000000C                          EXPORT HELLOWORLD_1
.text:1000000C              HELLOWORLD_1
.text:1000000C 10 40 2D E9              STMFD   SP!, {R4,LR}
.text:10000010 88 0F A0 E3              MOV     R0, #0x22C
.text:10000014 00 01 00 EB              BL      sub_1000041C
.text:10000018 00 40 50 E2              SUBS    R4, R0, #0
.text:1000001C 03 00 00 0A              BEQ     loc_10000030
.text:10000020 04 00 A0 E1              MOV     R0, R4
.text:10000024 D4 01 00 EB              BL      sub_1000077C
.text:10000028 08 30 9F E5              LDR     R3, =dword_10000E20
.text:1000002C 00 30 84 E5              STR     R3, [R4]
[...]
```

Java

Tout d'abord, il faut signaler que les risques liés à l'exécution d'un MIDLet Java « hostile » sont limités par la pauvreté de l'API CLDC/MIDP. Par exemple, l'envoi de SMS (cf. scénario 5) n'est pas prévu dans les spécifications de base et fait l'objet d'une JSR, ou d'une implémentation spécifique constructeur (cf. attaque sur le Siemens S55 ci-après).

De plus, des messages de confirmation sont présentés à l'utilisateur avant toute opération sensible, telle que l'émission d'un appel ou d'un SMS (même si l'expérience Internet Explorer a prouvé que les utilisateurs avaient une fâcheuse tendance à cliquer sur « Oui »).

Les principaux risques à considérer sont :

→ Des erreurs d'implémentation dans le modèle de sécurité de la machine Java (risque accru compte tenu des compromis imposés par la faible mémoire et la faible puissance de calcul des téléphones). Cf. bogue du Siemens S55 ci-après.

→ L'existence d'API « dangereuses », non identifiées comme telles, dans la norme de base ou dans les JSR. On notera par exemple avec intérêt la présence d'une API `platformRequest()`, qui rappelle furieusement les gestionnaires de protocoles présents dans Windows et MacOS X, sources de tant de problèmes récemment (`ms-its://`, `shell://`, `hcp://`, etc.). Par exemple, la syntaxe suivante est parfaitement valide :

```
platformRequest(«tel:012345678»);
```

À l'heure actuelle, il existe peu d'attaques publiées démontrant des failles dans les MIDLets Java, mais il est probable que de nombreux chercheurs travaillent sur le sujet.

Symbian OS

Possibilités d'exploitation malveillante

L'API Symbian est extrêmement riche [12]. D'autre part, les applications Symbian sont compilées directement en code assembleur ARM, ce qui leur confère rapidité et accès illimité à la plate-forme. Une application native Symbian peut virtuellement effectuer n'importe quelle opération sur le téléphone, à cause de l'absence de modèle de sécurité (pas de notions d'utilisateur, de liste de contrôle d'accès aux fichiers, etc.).

De plus, une grande partie de l'API se trouve dans des librairies et non dans le noyau, ce qui permet de contourner plus facilement les éventuels mécanismes de sécurité. Ainsi, pour envoyer un SMS, et en supposant qu'une confirmation utilisateur soit imposée par l'API, il suffirait d'appeler directement la fonction interne d'envoi de SMS plutôt que de passer par l'API standard pour contourner cette sécurité.

Limites et sécurités natives

Une limite importante, que personne n'a été en mesure de contourner jusqu'à présent, est la phase d'installation de l'application « hostile ». Sur le Nokia N-Gage par exemple, l'exécution directe d'un « .APP » est interdite « pour raisons de sécurité ». On peut légitimement se demander si le filtrage est effectué sur l'extension ou sur le contenu ...

De manière générale, l'installation d'un « .SIS » (package d'installation) nécessite au moins deux confirmations de l'utilisateur, sauf si celle-ci a été signée par une autorité reconnue par le téléphone. On peut craindre qu'un nombre non négligeable d'utilisateurs auraient la curiosité d'installer une application inconnue reçue par SMS ...

De nombreuses autorités sont pré-installées sur les téléphones. Sur le Sony Ericsson P900 par exemple, ce sont : Baltimore, Entrust, GTE CyberTrust, GlobalSign, RSA Data, Sony Ericsson, Symbian, Thawte et Verisign.

Personnellement, j'ai ainsi pu signer une application avec un certificat de l'autorité « *Thawte Personal Freemail Issuing CA* », obtenu gratuitement et sans fournir aucune information valide, à l'exception d'une adresse mail chez un WebMail. Le seul problème restant à résoudre est que l'autorité de délivrance du certificat (« *Thawte Personal Freemail Issuing CA* ») est une sous-autorité de « *Thawte Personal Freemail CA* », seule reconnue par le téléphone, et que la chaîne de confiance n'a pas pu être vérifiée par le téléphone (limitation de Symbian OS ou de l'outil MAKESIS ?). Tout cela est néanmoins de mauvais augure pour le modèle de sécurité...

Symbian envisage de mettre en place en 2005 un programme « Symbian Signed », qui impose la signature de toute application Symbian [13]. Toutefois, on ne connaît pas actuellement les solutions envisagées pour le problème des auteurs de *sharewares* et de *freewares*, qui n'ont pas les moyens financiers d'adhérer à ce programme.

Parmi les scénarios de risque présentés en introduction, de nombreux sont réalisables grâce à l'API Symbian. Prenons l'exemple de la lecture de l'IMSI (*International Mobile Subscriber Identity*) – scénario n°4. Sur le Sony Ericsson P900, l'IMSI se trouve dans le fichier `C:\System\data\imsi.txt`, lisible par n'importe quelle application ! (Ce code est d'ailleurs souvent utilisé dans les systèmes d'enregistrement d'applications, pour identifier de manière unique l'utilisateur).

Remarque : probablement pour d'obscures raisons de compatibilité, le système de fichiers Symbian est structuré de la même manière que... MS-DOS ! Ainsi, la ROM est-elle située sur le disque Z:, tandis que l'espace de stockage interne se trouve sur C: et les cartes mémoires additionnelles sur D:, E:, etc.

Windows CE

Les applications Windows CE présentent les mêmes fonctionnalités et les mêmes risques que les applications Symbian OS, avec une communauté de développeurs potentiellement plus importante puisque les API sont très proches du monde Windows « traditionnel » et donc le temps d'apprentissage très réduit.

Il n'est alors pas surprenant de constater que les attaques publiées ces derniers temps (cf. ci-après) affectent principalement les plates-formes Windows CE.

N'ayant pas la chance d'avoir un téléphone sous Windows CE à disposition, je laisse le sujet ouvert pour un éventuel futur article...

Quelques attaques réelles

Le bogue du S55

Lors de la conférence Black Hat USA 2003, le groupe Phenoelit révélait une faille de type « *race condition* » dans les téléphones Siemens S55, permettant d'envoyer un SMS *avant* l'affichage de la fenêtre de confirmation.

Bien que limitée à un seul type d'appareil, et donc difficilement exploitable « dans la nature », cette faille reste le meilleur exemple d'un bogue d'implémentation dans la machine Java ouvrant la voie à des attaques réelles, de type « envoi de SMS surtaxés » (scénario n°5).

De plus, il est intéressant de constater que ce bogue a été corrigé dans les *firmwares* les plus récents, sans qu'aucune information n'ait été fournie au public.

La correction d'une faille de sécurité dans le *firmware* d'un téléphone reste en effet à l'heure actuelle un problème largement insoluble, et compensé par le renouvellement très rapide du parc. Dans le cas du Siemens S55, il est nécessaire de posséder le câble de liaison PC pour mettre à jour le *firmware*, et cette opération s'effectue à l'initiative de l'utilisateur.

Vers Cabir et Dust

Les vers Cabir [14] et Dust [15], médiatisés à outrance par tous les éditeurs antivirus, ont été présentés comme les premiers codes malveillants pour téléphones portables et PocketPC. On notera toutefois qu'il n'existe aucune souche de ces vers en circulation, puisque ceux-ci ont été remis directement par leurs auteurs (des membres du groupe 29A[1] [16]) aux éditeurs antivirus !

Pour une analyse complète de ces vers, se reporter à l'article d'Eric Filiol dans ce même numéro, ou à l'analyse de l'auteur lui-même, puisqu'il a publié le code source sur Internet [17] (en anglais). Les problèmes liés à l'infection d'exécutables sous Windows CE mériteraient un article MISC à eux seuls : DLLs en ROM, absence d'en-tête sur les fichiers exécutables chargés en mémoire, etc.

L'astuce utilisée par l'auteur pour localiser `KERNEL32.DLL` en mémoire consiste à passer par la structure `KDataStruct`, qui se trouve à un emplacement fixe en mémoire (comme le PEB sous Windows NT) et qui contient une référence aux modules chargés (cf. `\PRIVATE\WINCEOS\COREOS\NK\INC\nkarm.h` dans le code source de Windows CE 4.2).

A l'heure actuelle, ces vers tiennent plus du « *proof-of-concept* » pour plusieurs raisons :

- → Le problème de la confirmation utilisateur n'a pas été résolu – il est donc possible de ne pas être infecté en cliquant simplement sur « non ».
- → La compatibilité multiplateforme n'est pas garantie.

Le risque associé est donc pour le moment extrêmement faible.

Trojan Brador

Le cheval de Troie « Brador » [18], s'exécutant sur les systèmes Windows CE, ouvre un port TCP et attend les ordres de son créateur.

Là encore, il s'agit d'un code transmis directement aux éditeurs antivirus, qui ne possède pas de capacité de réplication, et le risque associé est donc extrêmement faible.

On notera toutefois que le portage d'un cheval de Troie traditionnel sur Windows CE est une opération relativement simple, grâce à la proximité des API entre les plates-formes Windows.

Dialer Mosquito

Pour une fois, il s'agit là d'un code présent « dans la nature » bien que non intentionnellement malveillant !

Résumé des faits : l'application Mosquito, développée par la société Ojom, est disponible sur les réseaux « peer-to-peer » en version « crackée ». Toutefois, cette version n'est pas la version finale et contient une fonction d'envoi de SMS qui a été supprimée dans la version finale, d'où l'inquiétude légitime des utilisateurs voyant des SMS émis régulièrement à partir de leur téléphone portable !

Bien que cet exemple soit anecdotique, le communiqué de presse de la société Symbian [19] est tout à fait intéressant quant à leur vision de la sécurité. Celle-ci se résume en deux points :

- → L'installation d'antivirus sur les téléphones portables, en partenariat avec les éditeurs.
- → Le contrôle des développeurs, par la nécessité à terme d'obtenir une signature numérique pour toute application Symbian (programme « Symbian Signed »).

Malheureusement, il y a fort à parier que cette mesure ne détourne les développeurs indépendants de la plate-forme Symbian et ne cause à terme sa perte...

Quelques mots sur les failles BlueTooth

De nombreuses rumeurs circulent sur la sécurité du protocole BlueTooth et son utilisation comme vecteur de propagation par les virus, tel que Cabir.

En préambule, il faut souligner qu'à l'heure actuelle, il n'existe aucune attaque conceptuelle connue permettant de mettre en défaut la sécurité des communications établies via la norme BlueTooth. Nous

[1] Le groupe 29A (666 en hexadécimal) est un groupe de créateurs de virus très actif. Il publie entre autres un eZine annuel. **http://29a.host.sk/**

ne sommes pas dans le cas de la norme 802.11b (dite WiFi), où pour mémoire l'algorithme WEP peut être considéré comme cryptographiquement défaillant à cause, par exemple, du trop faible espace de vecteurs d'initialisation (cf. dossier MISC n°6).

On notera toutefois que la sécurité BlueTooth (via la mise en place d'un code PIN lors de l'appariement des équipements), tout comme la clé WEP, est un mécanisme optionnel : il est vivement recommandé de l'activer sur son téléphone !

Ceci étant dit, des problèmes semblent avoir été identifiés dans l'implémentation de la norme BlueTooth par les constructeurs.

Un chercheur travaillant pour la société Integralis sous le pseudonyme de « c0rnholio », s'est vanté d'avoir créé un outil baptisé « btchaos », qui permet d'exploiter une faille dans l'implémentation BlueTooth des téléphones Nokia 6310i et Sony Ericsson T610. Cette faille permet d'échanger des informations avec la cible, sans passer par la phase d'association BlueTooth.

Une des conséquences de cette attaque, si elle est avérée, est la possibilité d'émettre des SMS depuis un équipement BlueTooth proche via une commande modem « AT » traditionnelle. Il est ainsi possible d'accéder de manière frauduleuse à des services pilotés par SMS, tels que les *hotspots* WiFi des opérateurs [20].

L'alerte semble crédible puisque Nokia parle sur son site de mises à jour disponibles pour les téléphones concernés [21].

Adam Laurie avait déjà signalé une attaque de ce type, baptisée « BlueSnarfing », en novembre 2003 [22]. Malheureusement, là encore, aucun outil n'a été diffusé publiquement, et cette information reste difficile à vérifier. La presse rapporte toutefois que les conversations de plusieurs parlementaires britanniques auraient pu être écoutées par le biais de cette attaque [23].

Alors, info ou coup commercial ? Il semble qu'il y ait une part de vérité dans ces assertions.

Sans rentrer dans les détails de la norme BlueTooth [24], il n'existe actuellement que 13 profils standards, les autres étant en phase de standardisation plus ou moins avancée (profil = fonctionnalité dans le jargon BlueTooth).

Profils standards

K1	GAP	*Generic Access Profile*
K2	SDAP	*Service Discovery Application Profile*
K3	CTP	*Cordless Telephony Profile*
K4	IP	*Intercom Profile*
K5	SPP	*Serial Port Profile*
K6	HS	*Headset Profile*
K7	DNP	*Dial-up Networking Profile*
K8	FP	*Fax Profile*
K9	LAP	*LAN (Local Area Network) Access Profile*
K10	GOEP	*Generic Object Exchange Profile*
K11	OPP	*Object Push Profile*
K12	FTP	*File Transfer Profile*
K13	SP	*Synchronization Profile*

Parmi les autres profils qu'on rencontre sur le marché, citons : *Audio Gateway Profile* (AG), *Personal Area Network* (PAN), *Human Interface Device* (HID), *Hardcopy Cable Replacement Profile* (HCRP), etc.

Les propriétaires d'oreillette BlueTooth expérimentant des problèmes insolubles d'une marque à l'autre apprécieront la subtile différence entre le profil « Headset » et le profil « Audio Gateway »...

Les profils disponibles sur le téléphone sont énumérables via SDP (*Service Discovery Protocol*). Or certains téléphones possèdent des profils qui n'apparaissent pas dans cet annuaire, mais qui sont néanmoins accessibles, parfois sans authentification !

On peut supposer que ces profils sont utilisés dans les applications propriétaires des éditeurs, permettant la synchronisation Téléphone/PC par exemple, pour que l'utilisateur n'ait pas à taper de code PIN...

Des outils existent sous Linux, permettant de « bruteforcer » les profils de son téléphone. Un profil caché, permettant d'émettre des commandes AT et donc d'accéder au carnet d'adresses ou d'émettre des SMS sans authentification, a été découvert dans les téléphones Nokia 6310i, Ericsson T610 et T68i par exemple.

Le code est disponible sur le site suivant :
http://www.saftware.de/bluetooth/btxml.c

Quant aux outils et aux résultats obtenus sur quelques téléphones du marché, ils sont disponibles dans la *Bluetooth Device Security Database* [25].

Conclusion : mythe ou réalité ?

J'espère que cet article aura contribué à démystifier les menaces pesant sur les équipements mobiles (sans les minimiser), et aura permis au lecteur de se familiariser avec le développement sur ces plates-formes.

De manière générale, le portage des codes malveillants du monde PC sur des systèmes d'exploitation évolués tels que Symbian OS ou Windows CE ne posera aucun problème à un développeur expérimenté, et il est légitime de s'attendre à retrouver les mêmes problèmes dans le monde mobile que dans le monde PC, d'autant qu'un gain financier est possible (données à valeur commerciale telles que le carnet d'adresses, appels ou envois de SMS vers des numéros surtaxés, etc.).

Les pierres d'achoppement sont la propagation du code, qui nécessite une ou plusieurs confirmations de l'utilisateur pour s'activer sur la cible, et sa portabilité entre plates-formes. Gageons que ces problèmes risquent de trouver rapidement une solution ...

Remerciements
Cédric Blancher pour avoir fait don de son Siemens S55 à la science.
Eric Landuyt du support IDA pour son efficacité.

Références

Introduction

[1] Timifonica
http://www.secuser.com/alertes/2000/timofonica.htm
[2] SpyPhone ; **http://www.netespion.com/produits/phone.htm**
[3] HoaxBuster ; **http://www.hoaxbuster.com/hoaxliste/hoax.php?idArticle=1448**

Documentation sur les téléphones

[4] ARM Processors ; **http://www.arm.com/**
[5] ARM Chips List ; **http://www.geocities.com/~banksp/Archives/ARMChips.html**
[6] ARM Processor Programming Techniques
http://infoeng.ee.ic.ac.uk/~gac1/Architecture/Progtech.pdf
[7] The K Virtual Machine ; **http://java.sun.com/products/cldc/wp/**
[8] J2ME Wireless Toolkit ;
http://java.sun.com/products/j2mewtoolkit/
[9] Java API Documentation ; **http://java.sun.com/reference/api/**
[10] Symbian SDKs ;
http://www.symbian.com/developer/sdks.asp
[11] Symbian OS System Definition ; **http://www.symbian.com/developer/techlib/papers/SymbOS_cat/SymbianOS_cat.html**
[12] Symbian 7.0 API Reference ; **http://www.symbian.com/developer/techlib/v70docs/sdl_v7.0/doc_source/reference/cpp/**
[13] Symbian Signed ; **https://www.symbiansigned.com/**

Attaques réelles

[14] Ver Cabir ; **http://www.f-secure.com/v-descs/cabir.shtml**
[15] Ver Dust ; **http://www.bitdefender.com/bd/site/virusinfo.php?menu_id=1&v_id=282**
[16] 29A Virus Writers Group ; **http://29a.host.sk/**
[17] Details Emerge on the First Windows Mobile Virus (Part 3 of 3)
http://www.informit.com/articles/article.asp?p=337071
[18] Trojan Brador ; **http://securityresponse.symantec.com/avcenter/venc/data/backdoor.brador.a.html**
[19] Dialer Mosquito
http://www.symbian.com/press-office/2004/pr040810.html

Bluetooth

[20] Integralis Security Advisory - WLAN Hotspot Piracy through Identity Theft ; **http://www.integralis.co.uk/about_us/press_releases/2004/150604SA.html**
[21] Nokia : BlueTooth and Security
http://www.nokia.com/bluetoothsecurity
[22] Serious flaws in bluetooth security lead to disclosure of personal data ; **http://www.thebunker.net/release-bluestumbler.htm**
[23] UK politians bugged with BlueSnarf ; **http://news.mobile9.com/2004/04/uk-politians-bugged-with-bluesnarf/**
[24] Bluetooth : un profil pour une fonction ;
http://www.zdnet.fr/produits/materiels/assistants_personnels/guide/0,39022009,1001647,00.htm
[25] Bluetooth device security database ;
http://www.betaversion.net/btdsd/db/

Comment renforcer la sécurité des téléphones portables ?

1. Introduction

L'article de Nicolas Ruff revient sur les annonces médiatiques récentes autour des vers Cabir et Dust, et montre que les menaces pesant sur les téléphones portables sont réelles. Eric Filiol, pour sa part, s'intéresse au ver Cabir, et montre que, contrairement à ce qui a été dit, il ne s'agit pas d'une innovation, mais simplement d'une infection informatique classique sur une plate-forme inhabituelle. Ces deux articles tendent vers la même conclusion, à savoir que les problèmes de sécurité rencontrés dans les téléphones portables vont, à l'avenir, être de plus en plus nombreux, et de plus en plus proches des problèmes rencontrés dans l'environnement des ordinateurs personnels et professionnels.

Faut-il pour autant mettre en place les mêmes mécanismes de protection que sur les ordinateurs classiques, comme essaient déjà de nous le faire croire les vendeurs d'antivirus ? Pas si sûr que cela ! Dans le cadre de cet article, nous montrons qu'en exploitant davantage les spécificités matérielles et logicielles des téléphones portables, il est possible de renforcer la sécurité tout en permettant à l'utilisateur de télécharger de nouvelles applications, éventuellement non sûres.

Nous reviendrons dans un premier temps sur l'architecture d'un téléphone portable. Nous présenterons ensuite la sécurité des communications GSM, puis nous nous intéresserons à la sécurité des applications contenues dans les téléphones portables. Nous conclurons en proposant une approche pour renforcer la sécurité des applications des téléphones portables.

Nous ne nous intéresserons pas ici aux problèmes liés à la sécurité physique des mobiles, ce sujet pourra faire l'objet d'un article à part entière dans un futur numéro de MISC [NDLE: oui, oui encore :-].

2. Architecture des téléphones portables

Dans la suite de l'article, nous nous plaçons dans le monde du GSM. Schématiquement, un téléphone portable est composé d'un appareil de téléphonie mobile (qu'on appellera, par la suite, le mobile) dont le rôle principal est de prendre en charge les fonctions d'interconnexion au réseau de téléphonie mobile, et de la carte SIM (*Subscriber Identity Module*) assurant principalement les fonctions de sécurité nécessaires à la sécurité de l'interconnexion. Nous présentons ici brièvement l'architecture générique de la carte SIM et du mobile.

2.1 Architecture de la SIM

Au niveau matériel, l'architecture de la SIM repose généralement sur un microprocesseur CISC 8 bits cadencé à 4,77MHz, mais de plus en plus, sur un microprocesseur RISC 32 bits cadencé à 30, voire 100MHz. La SIM intègre également de la mémoire sous forme de ROM qui accueille essentiellement l'OS, d'EEPROM qui sert à stocker les applications et les données persistantes, et de RAM. Il est à noter que de plus en plus de SIM intègrent également un cryptoprocesseur dédié aux fonctions cryptographiques.

Au niveau de l'architecture logicielle, les systèmes d'exploitation (OS par la suite) peuvent être classés en deux familles : les OS propriétaires, généralement développés pour fournir un unique service (au minimum, les fonctionnalités décrites dans la spécification GSM 11.11 [1]), et les OS dits « multi-applicatifs », tels que JavaCard. Concernant les OS « multi-applicatifs », ils sont dits « ouverts » lorsque le téléchargement de nouvelles applications, et dans certains cas, la mise à jour de l'OS, est rendu possible.

Aujourd'hui, la tendance est à l'uniformisation des OS autour de JavaCard. L'OS JavaCard fournit une abstraction de type machine virtuelle Java. Les applications sont ainsi développées en langage JavaCard, une version « allégée » de Java. À l'image de Java, Sun est en charge de la spécification de JavaCard, tant au niveau des APIs qu'au niveau du langage [2]. Néanmoins, ces APIs sont génériques, et n'intègrent pas les fonctionnalités des SIM. L'ETSI a donc défini une API spécifique pour les SIM de type JavaCard (spécification GSM 03.19 [3]).

Intéressons-nous maintenant à la sécurité des informations/applications contenues dans la SIM. Une caractéristique fondamentale de la SIM est que l'ensemble de ces informations/applications appartient à l'opérateur (et non à l'utilisateur). Il en résulte que, suivant le modèle économique de la téléphonie mobile, la SIM est propriété exclusive de l'opérateur. Il est ainsi possible de s'appuyer sur des mécanismes de sécurité fortement centralisés pour garantir un haut niveau de sécurité, même lorsque l'OS est dit « ouvert ».

Parmi ces mécanismes, citons l'évaluation (au sens critères communs) et l'authentification (via une signature numérique du code) des applications. La sécurité physique de la SIM garantit par ailleurs que ces mécanismes ne peuvent être contournés.

En conséquence, les risques de failles de sécurité au niveau des SIM sont faibles. Il est donc raisonnable de penser que les applications contenues dans la SIM sont saines, et donc de considérer la SIM comme un outil cryptographique de confiance (au sens de PKCS#11 [4]).

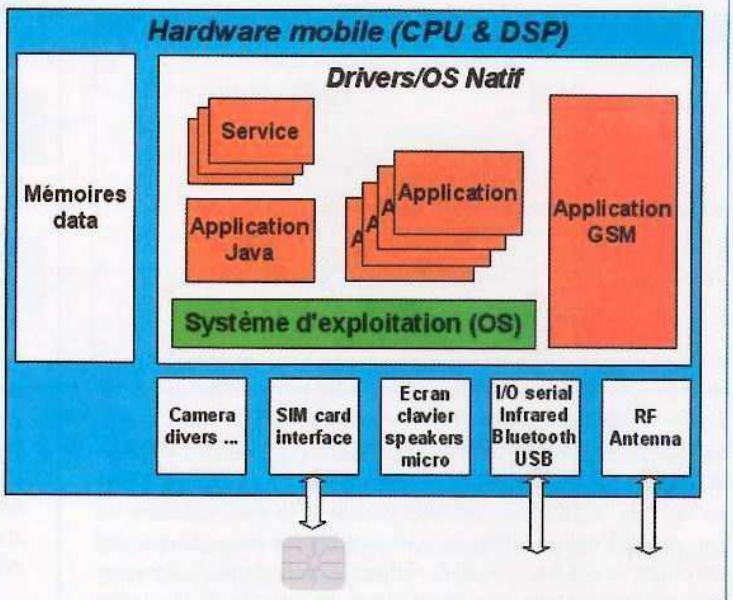
Nous ne discutons pas davantage de la sécurité de la SIM. En particulier, concernant la sécurité physique de celle-ci, nous renvoyons les lecteurs intéressés aux articles de Georges Bart publiés récemment dans MISC.

Christophe Bidan - **christophe.bidan@supelec.fr**
Supélec
Michel Morvan - **michel.morvan@mmce.mee.com**
Melco Mobile Communications Europe

2.2 Architecture du mobile

L'architecture matérielle du mobile repose généralement sur un processeur type RISC de la famille ARM dont le cadencement varie de quelques dizaines de MHz pour les mobiles entrée de gamme à plus de 100Mhz pour certains mobiles haut de gamme, la puissance du système étant intimement liée aux services proposés :Vidéo, Audio, Jeux, animations 2D/3D. Autour du CPU, on trouvera différentes mémoires : RAM et mémoire Flash avec des volumes supérieurs au méga-octet, parfois un peu de ROM et d'OTP pour des services dédiés à la sécurité. Des processeurs dédiés multimédia peuvent également être présents afin de compenser le manque de puissance de certains processeurs. Finalement, les interfaces écran, clavier, connecteurs (série, irda, BlueTooth), haut-parleur, micro, etc., habillent le tout. Ainsi, l'architecture matérielle est comparable à ce que l'on connaît dans le monde PC, avec bien souvent les mêmes fabricants. Néanmoins, il existe une différence importante entre les mobiles et les PCs, à savoir que la configuration matérielle d'un mobile n'est pas modifiable par l'utilisateur.

Figure 1 *Architecture logicielle du mobile.*



L'architecture logicielle est plus variable, suivant les fabricants de mobiles. Le choix de l'OS est globalement dicté par la stratégie commerciale, en particulier concernant le téléchargement de nouveaux services. Les OS propriétaires ne supportent généralement pas le chargement d'applications, à l'inverse des OS « ouverts » tels que Embedded LINUX [5], Symbian [6] ou encore WindowsCE [7]. Notons que la tendance actuelle va vers des mobiles à base d'OS ouverts, la mise à jour de l'OS étant par ailleurs possible dans certains cas.

L'ensemble des applications du mobile s'exécute au-dessus de l'OS (figure 1), à l'exception de l'application GSM, et des protocoles associés, qui, pour des raisons de performance, reposent généralement sur une implémentation propriétaire du fabricant du mobile, fortement liée au design hardware radio/fréquence de celui-ci. En vue de faciliter la portabilité des applications entre mobiles, les mobiles intègrent souvent en natif une machine virtuelle Java « allégée », J2ME (Java 2 Micro Edition). Cette machine virtuelle Java sert alors de plate-forme d'exécution pour les applications téléchargeables type jeux.

Tout comme pour la SIM, on pourrait penser à une tendance à l'uniformisation des OS. Il n'en est rien et la concurrence entre OS ouverts pour les parts de marché est importante, chaque OS cherchant à se distinguer par la nature/la qualité des « services » offerts.

Au niveau du mobile, une caractéristique essentielle est que, contrairement à la SIM, les informations/applications ne sont pas propriété exclusive de l'opérateur. Plus précisément, nous pouvons distinguer les informations/applications liées aux services de l'opérateur (par exemple l'application GSM), de celles liées à des services additionnels relevant de différents fournisseurs de services (par exemple, l'accès à un portail, e-commerce), ou encore de celles de l'utilisateur (par exemple un agenda électronique).

Les mécanismes de sécurité doivent prendre en compte les exigences de chaque acteur (voir paragraphe 4). Néanmoins, le modèle économique privilégie actuellement l'opérateur : il se positionne en tant que fournisseur de services vis-à-vis de l'utilisateur. À ce titre, il est à même d'assurer que le niveau de sécurité est conforme à l'attente de l'utilisateur. En particulier, dans le contexte d'un commerce électronique, l'opérateur garantit à l'utilisateur la confidentialité, l'authenticité et la validité de la transaction.

2.3 Interfaces entre le mobile et la SIM

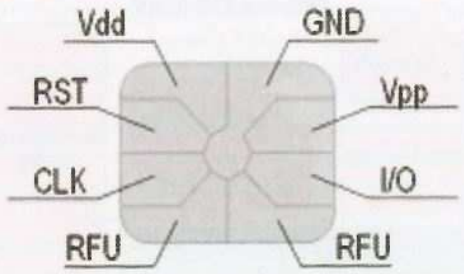
Cette partie s'applique à décrire la liste des interfaces courantes entre le mobile et la SIM.

La spécification GSM 11.11 de l'ETSI [1] définit les caractéristiques physiques (connecteurs – figure 2), électroniques (protocole de transmission) et logiques (commandes) de l'interconnexion du mobile et de la SIM. Cette spécification reprend en grande partie les éléments décrits dans la norme ISO/IEC 7816 [8] relative aux cartes à microprocesseur avec contacts, les différences portant essentiellement sur les algorithmes cryptographiques utilisés, les valeurs de certains paramètres, ou encore l'ajout de commandes spécifiques.

Au niveau applicatif, la spécification GSM 11.14 ou SIMToolKit de l'ETSI [9] définit une plate-forme pour le développement de

fonctionnalités implémentées dans la majorité des téléphones mobiles, une partie du code s'exécutant dans le mobile, l'autre au sein de la SIM. Parmi ces fonctionnalités, nous pouvons signaler la possibilité donnée à la SIM d'être « proactif », c'est-à-dire d'initialiser des actions sur le mobile sous forme de scripts ou d'applets JavaCard (affichage de texte, appel d'un numéro, établir une connexion GPRS data...). Ces applications SIM réalisent au besoin des services de sécurité en s'appuyant sur des clés et algorithmes de la SIM.

Figure 2 *Schéma du connecteur de la SIM*



La Spécification WIM (*WAP Identity Module* [10]) du OMA Forum [11] (anciennement WAP Forum) définit, pour sa part, une application au sein de la SIM avec une API dédiée à la sécurité dans les protocoles WAP. La SIM peut ainsi être utilisée pour stocker des certificats et des clés type RSA, réaliser les opérations de signature et d'authentification pour les services de sécurité de la couche transport (protocoles WTLS et TLS). Elle fournit également des services de sécurité applicative type WMLscript et ECMAscript permettant entre autres de déclencher une signature RSA dans la carte depuis une page WAP.

Comme nous l'avons signalé précédemment, les SIM sont de plus en plus souvent de type JavaCard. Dans le monde Java, une applet fait appel à une méthode d'une autre applet par RMI (*Remote Method Invocation*). Au niveau des JavaCard, nous retrouvons un mécanisme équivalent (JCRMI) pour accéder à une applet de la carte par simple appel de méthode, simplifiant encore l'interconnexion des applications du mobile avec celles de la SIM.

La spécification JSR 177 [12] étend cette approche aux fonctions de sécurité de la SIM. Plus précisément, la spécification JSR 177 définit un ensemble d'APIs pour accéder par RMI aux fonctions de sécurité implémentées dans la SIM qui est alors vue comme un outil de sécurité J2ME.

En résumé, l'ensemble de ces spécifications concourt au développement d'applications dont bien souvent la partie principale s'exécute dans le mobile, les services de sécurité étant exécutés dans la SIM. Nous pensons que cette fonctionnalité n'est, à l'heure actuelle, pas assez exploitée, alors qu'elle devrait être le principe de base de tout mécanisme de sécurité présent dans le mobile.

3. Sécurité GSM

Notre objectif ici n'est pas de présenter en détail la sécurité GSM, mais plutôt d'illustrer comment le mobile peut s'appuyer sur la SIM en vue de garantir un haut niveau de sécurité.

La fonction initiale des téléphones portables reste bien évidemment l'émission et la réception de voix et de données. En termes de sécurité, l'émission et la réception d'appels posent les problèmes d'accès au réseau de téléphonie mobile, de sécurité des communications, et pour l'opérateur, d'imputation des communications (lorsqu'il ne s'agit pas de communications prépayées).

Au niveau de l'accès au réseau de téléphonie, le contrôle porte sur l'authentification de la SIM, cette authentification étant réalisée de la manière suivante. Chaque SIM contient une clé secrète Kf, le « réseau » retrouvant Kf à l'aide d'une clé mère Km et des données d'identification de la SIM (ex.: numéro de série...). Le « réseau » envoie un challenge RAND, c'est-à-dire un nombre aléatoire de 128 bits. À la réception de RAND, la SIM calcule la réponse SRES sur 32 bits obtenue en appliquant un algorithme cryptographique appelé A3 (implémenté dans la SIM) au challenge RAND, avec la clé secrète Kf. Le réseau vérifie que la réponse SRES envoyée par la SIM correspond bien au challenge précédemment envoyé ; si tel est le cas, l'authentification de la SIM est valide, sinon l'authentification a échoué.

La sécurité des communications est obtenue en chiffrant les échanges entre le mobile et le réseau. Ce chiffrement nécessite la génération d'une clé de session. Cette clé de session Ks de 64 bits est obtenue en appliquant un algorithme cryptographique appelé A8 (implémenté dans la SIM) au challenge précédant RAND, avec la clé Kf. Une fois la clé de session Ks générée par le SIM, elle est fournie au mobile qui, à l'aide de l'algorithme cryptographique A5, va chiffrer les communications.

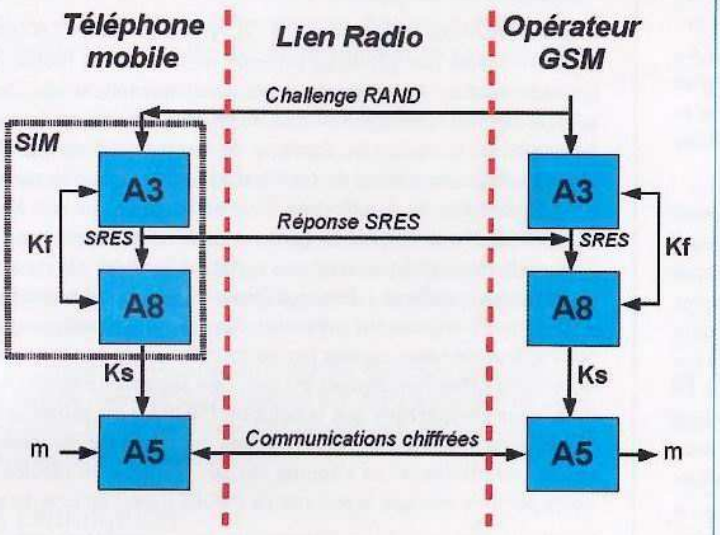
Le schéma global est résumé sur la figure 3. Nous voyons que la sécurité de l'authentification et de la génération de la clé de session repose exclusivement sur le secret de la clé Kf. Afin d'assurer un niveau de sécurité élevé, la clé Kf est stockée dans la SIM et n'est jamais communiquée au mobile (en particulier, les algorithmes A3 et A8 sont exécutés dans la SIM). Ceci est rendu possible dans la mesure où, rappelons-le, la SIM est propriété exclusive de l'opérateur : l'opérateur peut s'assurer que toute application contenue dans la SIM préserve la sécurité de Kf.

Concernant le chiffrement des communications, il repose sur le secret de la clé de session Ks. Or, la clé Ks est fournie au mobile qui, pour des raisons de performance, est en charge du chiffrement des communications. Il en résulte que, si le mobile est compromis (par exemple, via un virus), un attaquant peut avoir accès à la clé de session Ks, et ainsi déchiffrer l'ensemble des communications. Néanmoins, au regard des contraintes de performance, ce risque est inévitable. Il faut cependant prendre en compte ce risque lors du développement de l'application côté mobile, et, en particulier, contrôler l'utilisation des fonctionnalités sensibles afin d'empêcher un attaquant d'y accéder.

Signalons qu'en pratique, les algorithmes A3 et A8 sont généralement combinés. En particulier, l'ETSI a proposé les algorithmes (secrets) COMP128 (cassé en avril 1998), COMP128-2, et plus récemment, COMP128-3 pour réaliser l'authentification et la génération de la clé de session [16]. Notons que les opérateurs n'utilisent pas l'un de ces algorithmes, mais une version propriétaire.

Concernant l'algorithme de chiffrement A5, il est public. Les versions A5/2 (chiffrement faible) et A5/1 (chiffrement fort) ont fait l'objet d'attaques respectivement en août et en décembre 1999. En

Figure 3 Algorithmes A3/A8



revanche, il n'existe pas, à notre connaissance, d'attaque sur la version A5/3 de l'algorithme.

Intéressons-nous maintenant brièvement à l'imputation des communications. Le schéma adopté par les opérateurs repose sur le fait que la SIM est attribuée à un utilisateur donné. Toute communication est alors imputée à l'utilisateur porteur de la SIM authentifiée, ce dernier étant supposé s'être identifié via le code PIN. Notons cependant qu'il est donné à l'utilisateur le moyen de désactiver la vérification du PIN. Si l'utilisateur choisit cette option, en cas de vol de son mobile, toute communication sera imputée sur son abonnement, et ce malgré l'absence d'authentification du porteur de la SIM.

4. Sécurité des applications du mobile

Intéressons-nous dans un premier temps aux OS propriétaires fermés, c'est-à-dire ne supportant pas le téléchargement d'applications. Du point de vue d'un fournisseur de services, cela signifie soit que son service s'appuie exclusivement sur des services présents par défaut dans les mobiles, soit qu'il négocie un accord pour inclure le code nécessaire dans les mobiles.

Dans le premier cas, le constructeur du mobile pourrait (à la demande de l'opérateur) s'assurer, éventuellement via une évaluation au sens critère commun, que les services présents dans le mobile ne puissent être détournés à des fins malveillantes. Dans le second cas, l'accord devrait exiger l'évaluation du code avant installation, afin de vérifier qu'il ne contient ni porte dérobée, ni bogue résiduel. Dans les deux cas, le fournisseur de services ne serait pas en mesure d'introduire un malware dans le mobile.

Du point de vue de l'utilisateur, les OS propriétaires fermés rendent impossible le téléchargement de nouvelles applications. En d'autres termes, il n'est pas possible, pour l'utilisateur, de télécharger un malware en vue d'attaquer le mobile et/ou les services.

Intéressons-nous maintenant aux OS (propriétaires) ouverts, c'est-à-dire supportant au minumum le téléchargement de services ou de contenu via des applications type Java, mais aussi le téléchargement d'applications (généralement, en langage C) exécutées par l'OS, voire, dans certains cas, la mise à jour de l'OS lui-même. Dans ce contexte, le risque est grand de voir l'utilisateur installer, volontairement ou non, une application malveillante.

Du point de vue de l'attaquant, les objectifs visés sont :

→ de rendre inutilisable le mobile (temporairement ou définitivement) ;

→ de modifier le comportement du mobile ;

→ d'accéder aux données sensibles ;

→ d'émettre des appels voix ou données à l'insu de l'utilisateur ;

→ d'outrepasser les politiques de DRM (gestion de droits pour les contenus payants) ; ...

Conscients de ce problème, les constructeurs de mobiles proposent des solutions complémentaires dont les objectifs sont de garantir la sécurité tout au long du cycle de vie des applications. Schématiquement, le cycle de vie d'une application est composé des phases suivantes : le développement, le téléchargement, l'installation, l'exécution et l'exploitation.

Nous présentons les solutions actuellement déployées dans les différentes phases, ainsi que leurs limites.

4.1 Développement

Au niveau du développement, les langages utilisés sont généralement le C ou le langage Java. La programmation s'appuie en grande partie sur les APIs standards, mais il n'est pas rare de voir utiliser des APIs propriétaires afin d'optimiser le code, ou encore d'accéder à des services spécifiques. De telles pratiques augmentent les risques liés à la présence de failles de sécurité, ces APIs « contournant » les mécanismes de sécurité de l'OS.

Plus généralement, les problèmes de sécurité rencontrés lors de la phase de développement sont les mêmes que ceux rencontrés dans le monde PC, à savoir l'existence de bogues résiduels ou de portes dérobées, pouvant être exploités pour attaquer le mobile. Cependant, il existe une différence fondamentale dans le cadre des mobiles, à savoir que le code des applications est globalement moins important, rendant ainsi plus efficaces les techniques d'audit de code, ou encore possible l'évaluation, au sens critères communs.

Nous ne nous étendrons pas davantage sur cet aspect, considérant que les mécanismes de sécurité mis en place dans les phases suivantes ont pour objectif de précisément se protéger contre cela.

4.2 Téléchargement

Cette phase correspond au chargement d'une application dans le mobile à partir d'un serveur. Notons que le téléchargement peut être à l'initiative de l'opérateur (par exemple, pour une mise à jour logicielle), du fournisseur de services (par exemple, lors de l'accès à un portail), ou bien de l'utilisateur (par exemple, le chargement de nouveaux jeux payants pour la N-Gage).

Dans la phase de téléchargement, les problèmes de sécurité se posent au niveau de l'accès au service et du transfert de l'application. Concernant l'accès au service, les mécanismes à mettre en place sont côté serveur (contrôle d'accès par mot de passe, gestion des abonnements...) et ne rentrent pas dans le cadre de cet article. Au niveau transfert, il s'agit de garantir la confidentialité et/ou l'intégrité des échanges, afin d'empêcher un attaquant d'accéder au code et aux données de l'application, ou de modifier ce code et d'y introduire une faille.

À l'image de ce qui se fait sur Internet, l'utilisation de TLS/SSL pour sécuriser le niveau transfert nous semble une bonne solution. Ainsi, le WAP Forum a spécifié WTLS [13] qui n'est autre que l'implémentation de TLS dans WAP. Bien que généralement implémentée dans les mobiles, signalons cependant qu'une telle solution n'est pas souvent utilisée. Et lorsque c'est le cas, il est rare que l'ensemble des problèmes y afférant soit pris en compte. En particulier, la certification des clés publiques (pour empêcher une attaque de type « man-in-the-middle ») n'est généralement pas mise en œuvre, ou mal, ne tenant pas compte du fait que la notion de certification est encore moins compréhensible/intuitive pour un utilisateur mobile que pour un utilisateur PC. Dans la mesure où la possibilité de désactiver la sécurité au niveau transport est souvent donnée à l'utilisateur, l'absence de compréhension risque fort de se traduire par le choix de cette option.

En résumé, les solutions pour sécuriser le niveau transport, bien qu'existantes, ne sont généralement pas ou mal mises en œuvre. Les opérateurs préfèrent considérer que la sécurité au niveau transport est assurée par la sécurité « physique » du cœur de réseau, oubliant que ce dernier est de plus en plus souvent connecté à Internet.

4.3 Installation

Une fois l'application téléchargée, et avant son installation, il s'agit de vérifier qu'elle est saine, en particulier, qu'elle ne contient pas de virus et autres malwares. Ceci peut être réalisé soit par des antivirus, soit par des techniques de certification de code. La première solution n'est, à l'heure actuelle, que rarement utilisée, mais le tapage médiatique qu'a alimenté le ver Cabir nous laisse à penser que les distributeurs d'antivirus vont rapidement se lancer dans la course, à l'image de F-Secure qui a récemment annoncé qu'il proposera un antivirus aux acquéreurs du Nokia 6670.

Intéressons-nous plus particulièrement à la certification de code. Le principe fondamental est de signer numériquement le code afin de s'assurer de son authenticité. Actuellement, deux approches se distinguent : Symbian Signed [14] et MIDP2.0 [15].

Suivant l'approche Symbian Signed, toute application téléchargée doit être signée. Néanmoins, en l'absence de signature valide, l'utilisateur est prévenu, et la possibilité de continuer l'installation de l'application lui est donnée (message "*Unable to verify. Continue anyway ?*" pour Cabir), et ainsi, d'outrepasser le mécanisme de sécurité. En faisant précéder le téléchargement par un « faux » SMS prévenant de la mise à jour nécessaire du système, il est fort à parier que bon nombre d'utilisateurs continuera l'installation malgré le message de mise en garde.

Une fois installées, aucune distinction n'est faite entre les applications signées et les autres : il en résulte que tout malware ainsi installé a accès exactement aux mêmes fonctionnalités qu'une application légitime. Nous pensons donc que la solution retenue par Symbian Signed n'est pas satisfaisante.

Intéressons-nous maintenant à la solution MIDP2.0 pour environnement Java J2ME. Le principe de base est le même que précédemment, à savoir la signature numérique du code (c'est-à-dire des fichiers JAR) à base de PKI. Néanmoins, est associée à ce principe la notion de domaine de sécurité : un domaine de sécurité définit un niveau de certification qui est identifié par une ou plusieurs clés de certification. Tout code signé avec une clé de certification Kc est associé au domaine de sécurité identifié par Kc. Tout code non signé (ou avec une signature invalide) est associé à un domaine spécifique (*Untrusted Domain*). Il est ainsi possible de distinguer les applications présentes dans le mobile : celles signées par l'opérateur, celles signées par un fournisseur de service connu, ou encore celles non signées ou avec une signature invalide. À ce titre, nous considérons que la solution MIDP2.0 est satisfaisante. Néanmoins, précisons que la vérification des signatures est réalisée au sein du mobile, et ne s'appuie pas sur la SIM. Il en résulte, de notre point de vue, que la sécurité de MIDP2.0 peut être renforcée.

Signalons enfin, mais nous n'étudierons pas ce problème dans cet article, qu'une attention particulière doit être apportée aux applications soumises à droits (DRM – *Digital Right Management*) pour lesquelles il ne suffit pas de vérifier que l'application est saine, mais il faut également vérifier que l'utilisateur a acquis les droits l'autorisant à y accéder et sécuriser son installation.

4.4 Exécution

Lorsqu'elle est installée, une application est « exécutable ». À ce niveau, il peut sembler inutile de mettre en place des mécanismes de sécurité, dans la mesure où, pour être installée, l'application a dû être jugée saine. Néanmoins, une application saine ne veut pas dire qu'elle ne contient pas de portes dérobées ou de bogues résiduels. En outre, le pire ennemi de la sécurité est l'utilisateur à qui on donne encore une fois la possibilité d'outrepasser les mécanismes de sécurité, par exemple en lui demandant de confirmer une action jugée non sûre (message "*Unable to verify. Continue anyway ?*" pour Cabir). Il est donc préférable de mettre en place des mécanismes de sécurité afin de contrôler les actions des applications lors de l'exécution.

De manière générale, les mécanismes de sécurité à l'exécution dépendent fortement des mécanismes utilisés lors de l'installation. Ainsi, comme nous l'avons signalé précédemment, la solution Symbian Signed ne permet pas de distinguer les applications signées de celles non signées. Il n'est donc pas possible de contrôler à l'exécution les actions des applications non signées, qui peuvent ainsi accéder aux mêmes fonctionnalités que les applications signées.

En revanche, la notion de domaine de sécurité proposée dans MIDP2.0 est utile pour mettre en œuvre des solutions de contrôle à l'exécution. À l'image de la machine virtuelle Java, les principes de base consistent d'une part à vérifier que l'exécution est conforme au langage utilisé, et d'autre part, à contrôler l'accès aux ressources et aux données. Le premier point est couvert, dans la cas de MIDP2.0, par l'utilisation obligatoire d'un vérifieur de ByteCode. Pour le second point, à chaque domaine de sécurité est associé un ensemble de droits d'accès, ou privilèges, portant sur les données et les ressources du mobile. Pour une application donnée, son domaine de sécurité

définit donc l'ensemble des ressources et des données auxquelles elle est autorisée à accéder. En particulier, en définissant des droits très restreints pour les applications non signées ou à signature invalide, il est possible de les confiner dans un espace ne leur permettant pas d'accéder aux informations et services sensibles.

Bien entendu, lorsqu'ils sont implémentés, ces mécanismes de sécurité doivent être exempts de failles de sécurité. Cela signifie qu'ils doivent faire l'objet d'un audit poussé, ou d'une évaluation au sens critères communs. En particulier, la machine virtuelle Java doit être vérifiée en vue d'éliminer les éventuels bogues de programmation. À titre indicatif, signalons que, lors de la récente conférence « *Hack in the Box* » en Malaisie, des failles de sécurité dans J2ME ont été mises à jour, failles qu'un attaquant peut exploiter pour installer un malware.

En résumé, la solution de MIDP2.0 est, de notre point de vue, intéressante, mais doit s'accompagner par d'une évaluation du code. Par ailleurs, l'utilisation de la SIM permettrait d'élever encore le niveau de sécurité de la solution.

4.5 Exploitation

Certaines applications fournissent des services de sécurité que d'autres applications utilisent, à l'image d'applications fournissant l'API JSR177, ou bien WMLScript et ECMAScript pour WAP. Ces applications deviennent donc aussi sensibles que le code natif, et requièrent donc le même niveau d'exigence en termes de sécurité. Concrètement, dans le cadre de MIDP2.0, cela signifie que ces applications seront associées au même domaine de sécurité que le domaine du code natif.

4.6 Matériel

Notons que de nombreux constructeurs de mobiles utilisent, de leur propre initiative, des composants matériels spécifiques pour intégrer des services de sécurité tels que le stockage des clés ou des algorithmes de chiffrement et d'authentification. Outre le fait que ces composants font quelquefois doublon par rapport aux fonctionnalités présentes dans la SIM, cela se traduit par une augmentation du coût du mobile sans pour autant présenter de garantie quant à leur utilisation. Bon nombre de fournisseurs de sécurité (RSA, Certicom, Discretix…) proposent depuis quelques années déjà des briques matérielles de ce type aux constructeurs de mobiles.

5. Renforcer la sécurité des téléphones portables

Comme nous l'avons vu, la sécurité des applications de la SIM est depuis longtemps prise en compte. Suivant le modèle économique de la téléphonie mobile, la SIM est propriété exclusive de l'opérateur, ceci se traduisant par un contrôle absolu sur les applications et informations contenues dans la SIM. En s'appuyant sur les schémas d'évaluation et de certification, les encarteurs sont en mesure de garantir aux opérateurs le niveau de sécurité attendu. Il en résulte que la SIM peut être considérée comme un outil implémentant des services de sécurité de confiance.

La sécurité des applications du mobile est moins bien gérée. Plusieurs raisons concourent à cette moins bonne gestion :

- → la disparité dans les architectures matérielles des mobiles, et plus particulièrement au niveau du hardware dédié aux fonctions de sécurité ;
- → la disparité dans les architectures d'OS, et plus particulièrement au niveau des services de sécurité disponibles dans chaque OS ;
- → la diversité et la disparité des standards ;
- → la sous-exploitation des services sécurité de la SIM.

L'absence d'attaques à grande échelle n'incite pas les acteurs à prendre les mesures nécessaires. Malheureusement, encore une fois, il faudra attendre une telle attaque pour qu'enfin la sécurité soit vue comme un argument commercial, et donc prise en compte.

De notre point de vue, le renforcement de la sécurité des téléphones portables passe par la prise en compte des trois aspects suivants :

1 L'utilisation systématique des composants matériels dédiés sécurité :

- → la carte SIM (ou autre carte) pour tous les services d'authentification, de signature et de gestion de clés ;
- → les composants du mobile pour les opérations de cryptographie nécessitant de la performance (chiffrement symétrique par cryptoprocesseurs), pour le stockage des données sensibles et volumineuses, ainsi que pour la protection du code natif (en particulier, *boot loader* sécurisé par utilisation de mémoires ROM ou OTP) .

2 La sécurité du code qui réalise les opérations de sécurité ou les opérations sensibles (déclenchement SIMToolKit, envoi de SMS, présentation de PIN code) :

- → l'évaluation, au sens critères communs, du code ;
- → l'authentification du code en s'appuyant sur la carte ;
- → la protection du code en s'appuyant sur les composants dédiés au sein du mobile.

Ceci s'applique au code mobile natif (propriétaire et non modifiable), ainsi qu'au code mobile téléchargeable.

3 La gestion du niveau de sécurité des applications et services :

- → le cloisonnement des applications en s'appuyant sur la notion de domaines de sécurité telle que proposé par MIDP2.0 ;
- → la garantie du principe du « minimum nécessaire » : les applications ne doivent avoir accès qu'aux fonctions nécessaires pour rendre le service attendu ;
- → l'utilisation systématique des services de sécurité certifiés.

Dès lors que ces 3 aspects sont assurés, l'opérateur, l'utilisateur et les fournisseurs de services ont la garantie que toute opération de sécurité légitime s'appuiera sur des mécanismes fiables. Concrètement, ceci signifie pour l'utilisateur que ses secrets (clés et certificats) sont stockés dans la carte, et n'en sortent pas.

Inversement, les acteurs ont la garantie que toute opération illégitime ne mettra pas en danger la sécurité globale du téléphone portable. Ceci dé-responsabilise l'utilisateur : même si ce dernier accepte de télécharger et d'installer du malware, ce malware ne pourra faire autre chose que des opérations ne mettant pas en danger les services et applications sensibles du téléphone portable. En particulier, le malware ne sera pas en mesure d'accéder aux opérations utilisant les secrets contenus dans la carte.

Conclusion

Dans le cadre de cet article, nous avons cherché à analyser les raisons pour lesquelles la sécurité des téléphones portables n'est actuellement pas garantie. En résumé, nous pensons que la raison principale est un manque d'approche globale de la sécurité.

Ce manque d'approche globale s'explique par le nombre d'acteurs présents (l'encarteur, l'opérateur, les fournisseurs d'application, les constructeurs de mobiles et l'utilisateur), par la disparité des standards (même si des tentatives de convergence ont été faites, on ne peut pas dire aujourd'hui que ce point soit résolu), mais surtout par la concurrence que se livrent les constructeurs de mobiles.

Concrètement, cette concurrence se traduit par l'existence de plusieurs OS (Symbian, WindowsCE, Embedded Linux). Les utilisateurs devant pouvoir acheter et utiliser n'importe quel mobile du marché, les opérateurs se retrouvent donc devant l'impossibilité d'avoir un quelconque contrôle sur les applications « sensibles » contenues dans les mobiles. La conséquence est que certains opérateurs définissent leurs propres spécifications de mobiles afin de conserver le contrôle des services. Ceci est illustré aujourd'hui par l'offre de certains opérateurs majeurs tels que Vodafone (Vodafone Live !) et Docomo (IMode).

Côté utilisateur, ce manque d'approche globale se traduit par un risque important lors du téléchargement des applications, alors que, le mobile appartenant en premier lieu à l'utilisateur, il devrait être en mesure d'installer les applications qu'il souhaite. Certains nous diront que, à l'image de ce qui est fait dans Symbian, le téléchargement est possible, mais sous la seule responsabilité de l'utilisateur, cela pouvant aboutir, comme dans le cas de Cabir, à l'installation d'un malware. Nous répondons à cela que, de notre point de vue, c'est de la responsabilité du constructeur, de l'opérateur et du fournisseur d'OS de garantir que les fonctionnalités sensibles du mobile ne peuvent être utilisées à des fins malintentionnées.

Une approche globale à la sécurité des téléphones portables passe par la définition d'une architecture de sécurité intégrant le mobile, la SIM, les services de sécurité, et s'appuyant sur la certification du code des applications natives ou sensibles. Dans cet article, nous avons brièvement présenté les principes fondamentaux d'une telle architecture. Si un seul principe est à retenir, c'est que les services proposés par la SIM ne doivent pas se limiter au domaine de l'opérateur, mais bel et bien s'ouvrir aux champs applicatifs des OS ouverts.

Mais, soyons réalistes : la définition d'une telle architecture de sécurité nécessite de réunir autour d'une même table les grands acteurs de la téléphonie mobile (constructeurs de mobiles, encarteurs et opérateurs). Autant dire que les attaquants ont largement le temps de « s'amuser ».

Références

[1] ETSI – European Telecommunication Standards Institute, GSM 11.11 – Digital cellular telecommunications system (phase 2+). Specification of the Subscriber Identity Module – Mobile Equipement (SIM-ME) interface.

[2] **http://java.sun.com/products/javacard**

[3] ETSI – European Telecommunication Standards Institute, GSM 03.19 Module Application Programming Interface (SIM API); SIM API for JavaCard Stage 2

[4] PKCS #11: Cryptographic Token Interface Standard, **www.rsasecurity.com/rsalabs/node.asp?id=2124**

[5] Embedded Linux Consortium, **www.embedded-linux.org**

[6] Symbian, **www.symbian.com**

[7] Microsoft Windows Embedded, **www.microsoft.com/windowsce/**

[8] ISO 7816 - ISO/IEC 7816: Integrated Circuits Cards With Contacts.

[9] ETSI – European Telecommunication Standards Institute, GSM 11.14 - SIM ToolKit (STK) - Digital cellular telecommunications system (phase 2+), Specification of the SIM Application ToolKit for the Subscriber Identity Module – Mobile Equipement (SIM-ME) interface

[10] WAP-198, Wireless Identity Module.

[11] OMA Forum - Open Mobile Allaince, **www.openmobilealliance.org**.

[12] Java Community Process, Java Specification Request JSR177 Security and Trust Services API for J2ME (SATSA) 1.0, **www.jcp.org/en/jsr/detail?id=177**

[13] WAP-199, Wireless Transport Layer Security Specification.

[14] Symbian Signed, **www.symbiansigned.com**

[15] Mobile Information Device Profile, JSR37, JSR118, **http://java.sun.com/products/midp/**

[16] Applet Java implémentant le COMP128 - **http://www.riscure.com/CryptoTools/COMP128.html**

Programmation sécurisée : étude de cas

Cet article pourrait être la suite logique du dossier du numéro 12 de MISC qui portait sur la sécurité des logiciels. Nous nous concentrons ici sur l'étude de quatre logiciels réputés pour leur sécurité : Postfix, vsftpd, djbdns et Dovecot. Nous allons plus particulièrement nous intéresser à leur architecture, leur implémentation, mais aussi à la manière dont ils sont audités. Un grand merci aux auteurs - Chris Evans, Timo Sirainen et Wietse Venema - qui ont aimablement répondu à nos questions.

Introduction

Il y aurait beaucoup de choses à dire sur la manière de concevoir un programme informatique de façon sûre [1]. La plupart du temps, il s'agit juste de prendre en compte certaines recommandations, qu'elles soient dictées par la logique et le bon sens, mais également très techniques (et donc dépendantes d'un grand nombre de facteurs tels le système d'exploitation cible, le langage utilisé, le contexte d'exécution du programme, etc.).

Parfois, la conception d'un programme vraiment sûr ne peut se faire que grâce à la mise en œuvre de méthodes de développement logiciel particulières (par exemple les méthodes de spécifications formelles utilisées pour certains types de logiciels embarqués).

Suivant les choix effectués, les impacts sur la partie organisationnelle du projet peuvent devenir très importants.

Cet article vise à donner un aperçu général des moyens mis en œuvre dans quatre programmes open source conçus de manière sécurisée :

- vsftpd [2] (Chris Evans) est un serveur FTP considéré comme un des plus sûrs et des plus performants. Il est utilisé en production sur des sites à fort trafic.
- djbdns [3] (D.J. Bernstein) désigne un ensemble d'outils pour la mise en place des services DNS (serveur de cache, primaire, etc.).
- Postfix [4] (Wietse Wenema) est l'un des serveurs de messagerie (*Mail Transport Agent* - MTA) les plus utilisés. Son auteur l'a écrit comme une alternative à Sendmail plus simple à administrer, sécurisée et rapide.
- Dovecot [5] (Timo Sirainen) est un serveur POP3/IMAP dont le premier objectif est la sécurité. Mais ce n'est pas son unique qualité : il est très flexible, facilement configurable, peu gourmand en ressources et très rapide. Tous les exemples reposent sur la version stable 0.99.x.

Les caractéristiques de ces programmes sont résumées ci-après :

→ Le langage principalement utilisé est le C, terrain propice à bon nombre d'erreurs de programmation pouvant occasionner des failles de sécurité importantes.

→ Leur portabilité est excellente, et généralement les mécanismes de sécurité mis en œuvre ne sont pas spécifiques à un système Unix particulier.

→ Ils implémentent des *daemons* fournissant des services réseau. Les risques de compromission liés à une faille de sécurité exploitable à distance sont donc très élevés.

→ Enfin, ces différents services réseau s'appuient sur des protocoles Internet importants, souvent complexes à implémenter et contenant même parfois des problèmes de sécurité inhérents.

Les points abordés porteront à la fois sur les mécanismes de sécurité mis en place dans l'implémentation, mais également sur l'architecture, les méthodes de contrôle utilisées, etc. Nous n'entrerons pas dans le grand débat relatif à l'apport du mouvement Open Source en termes de sécurité, mais la caractéristique « logiciel libre » des programmes étudiés dans cet article sera à prendre en considération pour toute la partie relative à l'audit du code et plus généralement à la gestion du projet.

L'importance d'une architecture sécurisée

Pourquoi utiliser le terme d'architecture lorsque l'on parle de la conception d'un logiciel ? Tout simplement parce que l'expérience a montré que la réalisation de tâches complexes au moyen d'un programme dit monolithique n'a jamais donné de bons résultats en termes de sécurité.

Si on prend par exemple les tâches qu'un serveur de messagerie doit réaliser, cela ne paraît guère compliqué : acheminer les emails entrants vers les utilisateurs concernés du système et transmettre les mails sortants aux serveurs de mails directement responsables des domaines destination.

Maintenant, penchons-nous un peu (mais pas trop) sur les besoins que cela implique au niveau du système :

→ Le MTA doit réaliser des tâches nécessitant un niveau élevé de privilèges, comme accepter des connexions réseau sur le port TCP/25, avoir accès aux boîtes aux lettres de tous les utilisateurs et souvent la possibilité d'exécuter certains programmes sous leur identité.

Arnaud Guignard - **arno@rstack.org**
Pascal Malterre - **pascal@rstack.org**
Ingénieurs-chercheurs en sécurité des systèmes d'information au CEA-DAM

→ Il ne doit pas perdre de mails, et doit donc être conçu de la manière la plus robuste possible pour faire face à tous les types d'erreurs, depuis les problèmes matériels jusqu'aux erreurs plus complexes liées au protocole SMTP.

→ Il doit envoyer des requêtes DNS, ouvrir des connexions réseau vers les autres MTA, souvent fournir un moyen aux utilisateurs d'injecter localement des mails dans la file d'attente.

→ Avec l'évolution de certains problèmes liés à la messagerie, il doit pouvoir exécuter en amont toute sorte de programmes externes, comme pour faire de l'analyse de contenu (spam, virus, etc.)

→ Enfin, un serveur de mail **doit** être conçu en tenant compte de la sécurité pour différentes raisons : il travaille en permanence sur des données issues de sources non sûres, doit garantir une confidentialité minimale des messages entre les utilisateurs, etc.

Même sans trop entrer dans les détails, on comprend mieux qu'un programme conçu de façon monolithique ait beaucoup de mal à répondre à des exigences aussi diverses tout en garantissant une sécurité minimale. Dans le cas d'un MTA, on se rappellera le lourd passif de sendmail en termes de failles de sécurité.

Dans un monde idéal dépourvu de bogues, la conception d'une architecture sécurisée aurait beaucoup moins d'importance. Malheureusement, même les programmeurs les plus chevronnés ne sont pas à l'abri d'une erreur et si la plupart d'entre elles peuvent être détectées par l'audit régulier du code source, le risque zéro n'existe pas. La mise en œuvre d'une architecture sécurisée est un moyen permettant de limiter l'impact d'une faille de sécurité sur l'ensemble du logiciel, et souvent par voie de conséquence sur le reste du système.

Principe du moindre privilège et séparation des privilèges

Définition du principe du moindre privilège : chaque utilisateur et chaque programme doit disposer des privilèges minimaux pour mener à bien les actions qu'il doit effectuer et ce, afin de limiter les dommages en cas d'erreur, d'accident ou d'attaque.

Ce principe s'applique à différents niveaux :

→ Si un programme n'utilise pas de fonctions privilégiées ou bien si l'on peut faire en sorte qu'il n'ait besoin d'aucun privilège (sans remettre en cause son fonctionnement, bien sûr), alors il n'y a aucune raison de le concevoir en partant du fait que les droits root sont une condition nécessaire à son fonctionnement. Cela peut paraître évident, mais cela ne fait pas de mal de le rappeler :-)

→ Si un programme n'a besoin de privilèges que pour effectuer quelques actions ponctuelles bien définies, alors ces dernières doivent être regroupées et traitées le plus tôt possible lors de l'exécution du programme : on limite ainsi au maximum dans le temps l'exécution de code privilégié. Le programme `dnscache` illustre parfaitement ce principe :

```
-- ~/djbdns-1.05/dnscache.c --

int main()
{
  char *x;
  unsigned long cachesize;

  x = env_get("IP");
  if (!x)
    strerr_die2x(111,FATAL,"$IP not set");
  if (!ip4_scan(x,myipincoming))
    strerr_die3x(111,FATAL,"unable to parse IP address ",x);

  udp53 = socket_udp();
  if (udp53 == -1)
    strerr_die2sys(111,FATAL,"unable to create UDP socket: ");
  if (socket_bind4_reuse(udp53,myipincoming,53) == -1)
    strerr_die2sys(111,FATAL,"unable to bind UDP socket: ");

  tcp53 = socket_tcp();
  if (tcp53 == -1)
    strerr_die2sys(111,FATAL,"unable to create TCP socket: ");
  if (socket_bind4_reuse(tcp53,myipincoming,53) == -1)
    strerr_die2sys(111,FATAL,"unable to bind TCP socket: ");

  droproot(FATAL);
  ...
```

Une fois les deux sockets ouvertes en écoute sur les ports TCP/53 et UDP/53, le programme abandonne immédiatement ses privilèges en appelant la fonction `droproot()`. Cette dernière fait un `chroot()` (restriction de la visibilité de l'arborescence à un répertoire spécifique du processus), puis un `setgid()/setuid()` vers un compte utilisateur spécifique à dnscache.

→ Certains types de logiciels doivent en permanence exécuter des actions hautement privilégiées. On trouve par exemple dans cette catégorie des programmes serveurs `fork()ant` en permanence des processus fils s'exécutant sous des comptes utilisateurs légitimes du système (serveur SSH, etc.).

Dans ce cas, il faut appliquer le principe de séparation des privilèges et ainsi isoler les actions privilégiées dans des sous-systèmes bien spécifiques. Le nombre de tâches effectuées par ces derniers devra être réduit au strict minimum (typiquement les traitements des données issues de l'utilisateur ne devront jamais s'exécuter dans le même contexte). Enfin, les informations provenant des canaux de communication avec les autres processus (de niveaux de privilèges différents) devront être considérées comme non sûres et donc scrupuleusement vérifiées.

La mise en œuvre dans Dovecot

L'architecture de Dovecot est bâtie autour de quatre processus qui se partagent le travail suivant les privilèges nécessaires.

- `imap-master` fonctionne en root. Il se charge d'exécuter de nouveaux processus.

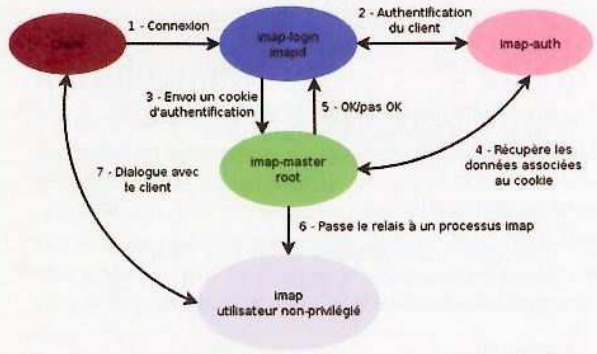
■ `imap-login` s'exécute sous l'identité d'un utilisateur non-privilégie (`imapd`). Il accepte les connexions et analyse les commandes envoyées par le client jusqu'à l'authentification.

■ `imap-auth` fonctionne avec l'identité de l'utilisateur dont il a besoin pour réaliser l'authentification (par exemple, si on utilise les *shadow passwords*, il aura besoin des privilèges root). Il dialogue avec `imap-master` et `imap-login` pour réaliser l'authentification.

■ `imap` fonctionne avec l'identité d'un utilisateur non-privilégié (celui à qui appartient le compte `imap`) et se *chroote* dans le répertoire de cet utilisateur. Il implémente toutes les commandes IMAP/POP3 et c'est avec lui que dialogue l'utilisateur une fois qu'il est authentifié.

Les échanges sont résumés dans la figure 1.

Figure 1 *Architecture de Dovecot*



En allant plus loin dans le principe de la séparation des privilèges, on peut mettre en place un cloisonnement bien plus strict que celui consistant simplement à isoler les portions de code nécessitant les droits root.

Par exemple, on peut isoler les différents sous-systèmes en fonction de leurs besoins en termes de ressources (accès à une partie du système de fichiers, au réseau, écriture des logs, etc.). La mise en œuvre d'une telle architecture est facilitée par les différents moyens de communication inter-processus disponibles pour les systèmes Unix.

L'exemple de Postfix

Postifx propose également un tel cloisonnement. Ce dernier est composé de plusieurs programmes chargés de différentes tâches :

■ `smtpd(8)` est le serveur écoutant sur le port TCP/25, il est chargé du dialogue SMTP.

■ `cleanup(8)` traite les messages avant de les placer dans la queue : il ajoute les en-têtes manquants, réécrit les adresses (via `trivial-rewrite(8)`), etc.

■ `qmgr(8)` est le *queue manager*. Sa tâche est de définir la manière dont va être distribué le mail.

■ `local(8)`, `smtp(8)`, `virtual(8)`, `pipe(8)`, etc., sont les agents de distribution du mail suivant sa destination, respectivement : pour un utilisateur local, pour un autre serveur de mail, pour un utilisateur dans un domaine virtuel, pour un programme, etc.

■ `master(8)` est le superviseur de tous les démons (invoqué lors du démarrage de Postfix avec `postfix start`). Il lance tous les processus indispensables au bon fonctionnement et vérifie qu'ils sont toujours actifs, les relançant si nécessaire.

Le comportement de tous ces processus est défini dans le fichier `/etc/postfix/master.cf` où l'on peut notamment spécifier si un programme doit être *chrooté* ou fonctionner avec un utilisateur non-privilégié (postfix si l'on a suivi les directives d'installation fournies). Un extrait du fichier par défaut est fourni ci-dessous :

```
-- /etc/postfix/master.cf --

#
==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
#
==========================================================================
smtp      inet  n       -       n       -       -       smtpd
pickup    fifo  n       -       n       60      1       pickup
cleanup   unix  n       -       n       -       0       cleanup
qmgr      fifo  n       -       n       300     1       qmgr
rewrite   unix  -       -       n       -       -       trivial-rewrite
bounce    unix  -       -       n       -       0       bounce
...
```

On peut noter qu'à l'exception des programmes `pipe(8)`, `virtual(8)` et `local(8)`, tous les autres processus peuvent être non privilégiés et *chrootés*.

Comment définir et limiter les privilèges ?

Sous Unix, les privilèges d'un processus sont généralement déterminés par une multitude de paramètres. On trouve :

1 Les différents couples (UID, GID) associés au processus :

■ RUID (*Real User ID*), RGID (*Real Group ID*) : il s'agit de l'UID et du GID du compte utilisateur sous lequel le processus s'exécute.

■ EUID (*Effective User ID*), EGID (*Effective Group ID*) : ils sont utilisés par le système d'exploitation à chaque vérification des permissions du processus.

■ SUID (*Saved User ID*), SGID (*Saved Group ID*) : ils sont utilisés comme sauvegarde des RUID, RGID en cas de changement. Un processus peut ainsi abandonner **temporairement** ses privilèges en utilisant les appels système `seteuid(2)` et `setreuid(2)`. On notera que l'appel à `setuid(2)` fixe la même valeur à tous les UIDs (*real*, *effective* et *saved*) et donc qu'à partir de là, il n'est plus possible de regagner ses privilèges.

■ Les groupes Unix supplémentaires auxquels est rattaché l'utilisateur.

■ FSUID (*FileSystem User ID*), FSGID (*FileSystem Group ID*) : ils sont spécifiquement utilisés pour les vérifications d'accès au système de fichier (disponible uniquement sous Linux).

2 La visibilité du système de fichier dont dispose le processus, ou plus exactement la racine du système de fichier utilisée par le processus. Cette dernière peut être modifiée au moyen de l'appel

système `chroot(2)`. La mise en place d'une telle prison réduit considérablement les risques en cas de compromission, mais elle n'est pas invulnérable [13].

3 Les *capabilities* POSIX sont un moyen de partitionner l'ensemble des privilèges root en un ensemble de privilèges bien distincts [6]. Bien que définies par POSIX, les spécifications sont restées à l'état de draft, et elles ne sont donc pas disponibles sur tous les systèmes Unix.

4 Les limitations par processus gérées par le système au moyen des appels `setrlimit(2)` et `getrlimit(2)`.

La séparation des privilèges dans vsftpd

La mise en œuvre de la gestion des privilèges à l'aide des fonctionnalités exposées ci-dessus est très claire dans le code de vsftpd. De plus, des quatre programmes que nous avons étudiés, il est le seul à faire usage des capabilities (à condition bien sûr qu'elles soient présentes sur le système).

Pour illustrer le fonctionnement, nous allons tracer l'exécution du programme depuis son lancement jusqu'à l'acceptation d'une première connexion FTP et essayer de bien comprendre le rôle des différents processus.

1 Différentes tâches préliminaires sont effectuées dans la fonction `(main.c):main()` : lecture du fichier de configuration, initialisations diverses, ouverture des sockets en écoute si on n'utilise pas de *tcpwrapper*, etc.

2 La fonction `(twprocess.c):vsf_two_process_start()` est ensuite appelée et prend en charge le déroulement de la suite des opérations.

a) Un canal de communication privé est créé au moyen de l'appel système `socketpair(2)`. Il sera utilisé pour l'échange de messages entre le processus père (privilégié) et le processus fils (non privilégié).

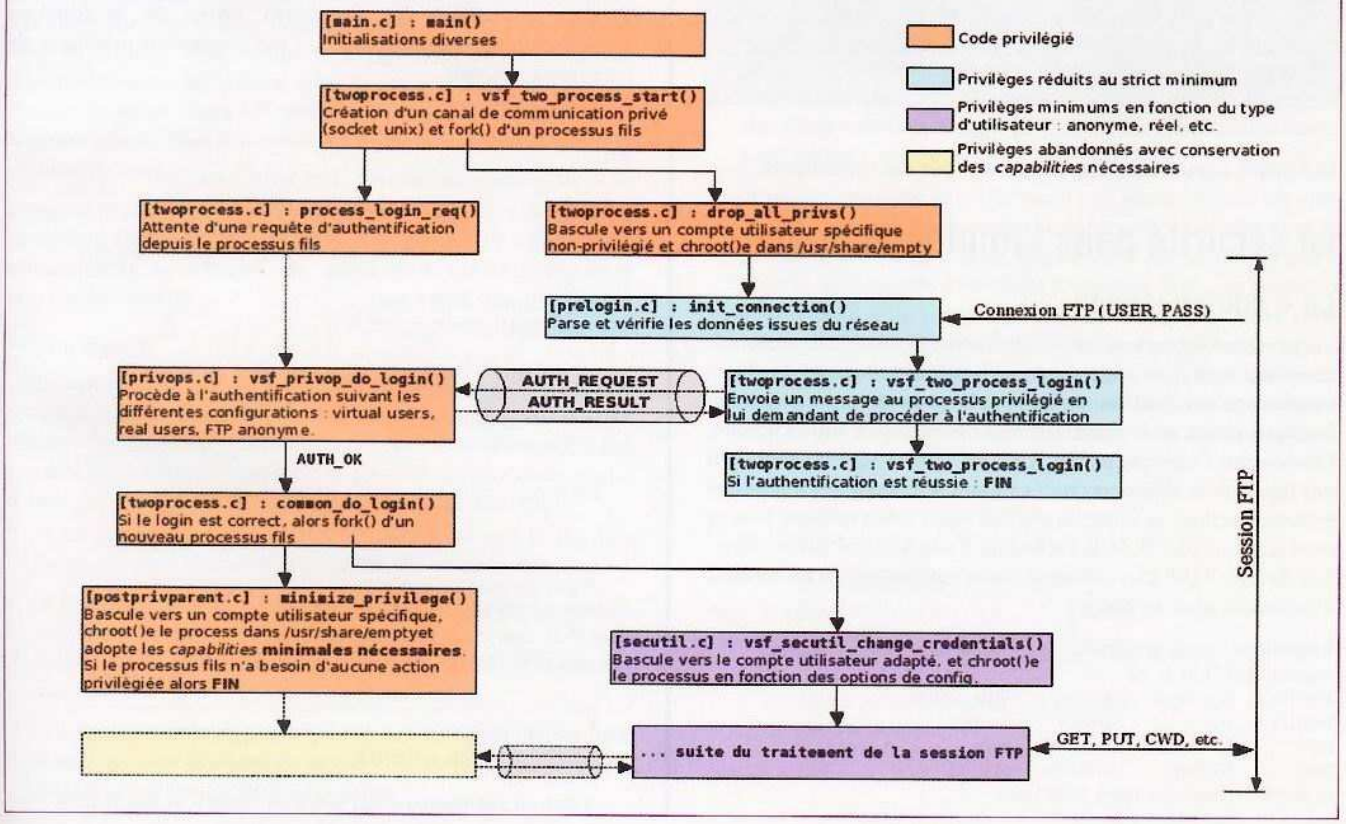
b) Un nouveau processus est créé au moyen d'un `fork(2)`

i) Le père appelle la fonction `(twoprocess.c):process_login_req()` et se met en attente des premières demandes de connexion transmises par une des extrémités du canal de communication créé en 2.a

ii) Le fils appelle la fonction `(twoprocess.c):drop_all_privs()`, puis attend les premières connexions dans la fonction `(prelogin.c):init_connection()`.

Au moment de l'appel à `drop_all_privs()`, le processus fils dispose donc encore des pleins pouvoirs sur le système. Cette fonction récupère le nom de l'utilisateur non-privilégié spécifié dans le fichier de configuration, vérifie que le répertoire destination du chroot est accessible et appelle `(secutil.c):vsf_secutil_change_credentials()`.

Figure 2 *Séparation des privilèges dans vsftpd*



3 La fonction `(prelogin.c):init_connection()` analyse la requête de connexion reçue par le réseau. Elle appelle ensuite la fonction `(twoprocess.c):vsf_two_process_login()` chargée de transmettre la demande de login au travers de la socket vers le processus privilégié. Si le résultat est OK, le processus fils n'a plus de raison d'être et met tranquillement fin à ses jours au moyen d'un `_exit(0)`.

4 Le processus père traite la requête de connexion et, en cas de succès, *forke* un nouveau processus fils dont les privilèges sont encore une fois réduits au minimum au moyen de `(secutil.c):vsf_secutil_change_credentials()`. Le processus père appelle enfin `(postprivparent.c):vsf_priv_parent_postlogin()`.

5 À ce stade, on dispose donc d'un processus fils créé en 4 et d'un processus s'exécutant en root chargé d'effectuer certaines actions privilégiées que le fils n'a pas le droit de faire.

Enfin, la fonction `minimize_privilege()` permet au processus père de conserver uniquement les privilèges dont il a besoin, calculés dynamiquement à partir des informations de configuration :

→ Dans le meilleur des cas, il n'y a plus aucune action privilégiée à effectuer, et le père peut donc se terminer sereinement.

→ Dans le pire des cas, il bascule vers l'identité de l'utilisateur non privilégié en conservant juste les capabilities nécessaires, puis se *chroote* dans un répertoire vide !

6 La suite de la session FTP est alors entièrement prise en charge par le processus fils excepté éventuellement pour les quelques opérations privilégiées pour lesquelles il fera appel au père afin de les réaliser.

La figure 2 page précédente récapitule tous ces traitements.

La sécurité dans l'implémentation

Le « coding-style »

L'organisation du code et son écriture sont aussi des facteurs décisifs dans l'écriture d'un code sécurisé. Il faut être capable de situer simplement une fonction parmi tout le code et comprendre son fonctionnement en la lisant, sans avoir recours à d'autres fichiers. Dovecot est l'exemple parfait : tous les fichiers sources sont dans des répertoires différents classés suivant la fonction principale. Les fichiers, fonctions et variables ont des noms très explicites. Tout ça rend d'autant plus facile la recherche d'une fonction particulière : lors du login IMAP d'un utilisateur quelle fonction envoie les données d'authentification au client ?

Regardons les répertoires :

```
~/dovecot-0.99.10.9$ ls src
Makefile.am  imap-login/   lib-index/     login-common/
Makefile.in  lib/          lib-mail/      master/
auth/        lib-charset/  lib-settings/  pop3/
imap/        lib-imap/     lib-storage/   pop3-login/
Le répertoire imap-login semble prometteur :

~/dovecot-0.99.10.9$ ls src/imap-login
Makefile.am  client-authenticate.c  client.c  common.h
Makefile.in  client-authenticate.h  client.h
Et dans le fichier client-authenticate.c on trouve la fonction désirée :

-- src/imap-login/client-authenticate.c --

static void client_send_auth_data(struct imap_client *client,
                                  const unsigned char *data, size_t size)
{
        buffer_t *buf;

        t_push();

        buf = buffer_create_dynamic(data_stack_pool, size*2, (size_t)-1);
        buffer_append(buf, «+ «, 2);
        base64_encode(data, size, buf);
        buffer_append(buf, «\r\n», 2);

        o_stream_send(client->output, buffer_get_data(buf, NULL),
                      buffer_get_used_size(buf));
        o_stream_flush(client->output);

        t_pop();
}
```

La clarté de l'organisation des sources et la lisibilité du code sont des facteurs très importants en termes de sécurité, car cela diminue le risque d'erreur et facilite les tâches de relecture, audit, etc.

À ce propos, le nombre de lignes de code source représente-t-il vraiment un indicateur important ? En effet, nombre de personnes ont l'habitude d'exécuter un `cat *.c | wc -l` afin d'avoir une idée grossière de la complexité d'un programme. On peut citer le contre-exemple de vsftpd qui utilise un style très verbeux (condition obligatoire d'après l'auteur pour arriver à un programme très sûr). Voici par exemple les premières lignes de la fonction `vsf_secutil_change_credentials()` qui change les privilèges de l'utilisateur courant.

```
-- ~/vsftpd-2.0.1/secutil.c --

void
vsf_secutil_change_credentials(const struct mystr* p_user_str,
                               const struct mystr* p_dir_str,
                               const struct mystr* p_ext_dir_str,
                               unsigned int caps, unsigned int options)
{
  struct vsf_sysutil_user* p_user;
  if (!vsf_sysutil_running_as_root())
  {
    bug(«vsf_secutil_change_credentials: not running as root»);
  }
  p_user = str_getpwnam(p_user_str);
  if (p_user == 0)
  {
    die2(«cannot locate user entry:», str_getbuf(p_user_str));
  }
  {
    struct mystr dir_str = INIT_MYSTR;
    /* Work out where the chroot() jail is */
    if (p_dir_str == 0 || str_isempty(p_dir_str))
    {
      str_alloc_text(&dir_str, vsf_sysutil_user_get_homedir(p_user));
    }
    else
    {
      str_copy(&dir_str, p_dir_str);
    }
    /* Sort out supplementary groups before the chroot(). We need to access
```

```
     * /etc/groups
     */
    if (options & VSF_SECUTIL_OPTION_USE_GROUPS)
    {
      vsf_sysutil_initgroups(p_user);
    }
    else
    {
      vsf_sysutil_clear_supp_groups();
    }

...
```

L'implémentation à proprement parler

Timo Sirainen, l'auteur de Dovecot, explique [7] comment doivent être pensés les logiciels. Pour résumer la simplicité est la clé : KISS (*Keep It Simple, Stupid*). Pourquoi faire des vérifications de sécurité dans tout le code alors qu'il suffit que ces vérifications soient concentrées dans une API ? L'audit devient alors plus simple, car il se focalise dans l'étude de quelques API (sans négliger le reste du code bien entendu).

Dovecot

D'une manière générale, Timo Sirainen dispense quelques conseils pour écrire des programmes en C de manière sécurisée et simple [9]. Ce *mini-howto* n'est pas spécifique à Dovecot [10] et peut (devrait !) s'appliquer à tous les logiciels.

Prévenir les débordements de tampon

Pour cela, il suffit de ne jamais écrire directement dans les buffers, mais plutôt de passer par des API qui, elles, se chargeront de vérifier que les données sont de la bonne taille et/ou augmenteront la taille du buffer si nécessaire. Ainsi, au lieu de se poser des questions quant à la vérification de la taille d'une chaîne/d'une donnée/... tout au long du code, il vaut mieux créer une API pour toutes les manipulations sur les buffers, qu'ils soient composés de texte ou d'autres données. Cette API devra être auditée avec précaution et facilitera grandement la production de code une fois qu'elle sera prouvée robuste.

Dovecot implémente une API pour gérer les buffers de manière générique (`src/lib/buffer.h`) et deux autres API dédiées à la manipulation de chaînes de caractères (`src/lib/str.h` et `src/lib/strfuncs.h`).

Pas de free(3)

L'allocation de la mémoire est toujours un casse-tête pour le programmeur. L'utilisation combinée de `malloc(3)`/`free(3)` peut se révéler très dangereuse aussi bien pour les fuites de mémoire (on a oublié un `free(3)`...) que pour la sécurité (exploitation via un double `free(3)` par exemple). Diverses solutions existent :

→ Le passage par des buffers statiques comme le fait D.J. Bernstein pour le traitement des chaînes de caractères.

→ L'implémentation d'un ramasse-miettes (*garbage collector*) présent dans de nombreux langages de haut niveau (OCaml, Python, Java, Lisp,etc.). Cette solution est de loin la meilleure, mais une telle implémentation n'est pas simple.

→ La gestion de la mémoire grâce à des fonctions de plus haut niveau qui sont chargées de prendre en compte les `malloc(3)`, `realloc(3)` et `free(3)` correspondants.

Dans le cas de Dovecot, Timo Sirainen a choisi la dernière option et utilise deux types de données :

→ le *data stack* (`src/lib/data-stack.h`) : il a ré-implémenté une pile que l'on peut utiliser pour l'affectation de données temporaires. Le principe : on alloue lors du premier appel un bloc de mémoire de 32 Ko qui est subdivisé en 32 pages. Si l'on a besoin de plus de 32 pages ou de plus de mémoire, on alloue alors un nouveau bloc, etc. L'utilisation dans le code se fait de manière quasi-transparente : on récupère d'abord une page dans la pile en invoquant `t_push()`, puis on peut allouer des variables dans cette page et, à la fin de l'utilisation des données, on appelle `t_pop()` pour « désallouer » la page. Ces deux déclarations figurent en début et fin de nombreuses fonctions, comme par exemple celle que nous avons vue précédemment :

```
-- src/imap-login/client-authenticate.c --

static void client_send_auth_data(struct imap_client *client,
                                  const unsigned char *data, size_t size)
{
    buffer_t *buf;

    t_push();

    buf = buffer_create_dynamic(data_stack_pool, size*2, (size_t)-1);
    buffer_append(buf, «+ «, 2);
    base64_encode(data, size, buf);
    buffer_append(buf, «\r\n», 2);

    o_stream_send(client->output, buffer_get_data(buf, NULL),
                  buffer_get_used_size(buf));
    o_stream_flush(client->output);

    t_pop();
}
```

Les avantages sont nombreux : rapide, pas de `free(3)` pour chaque allocation (un seul suffit pour toute la page), pas de fuite de mémoire, pas de fragmentation. Le problème principal est que l'on peut par inadvertance affecter une variable qui a été allouée dans une page et accéder à cette variable après avoir fait un `t_pop()`.

→ les *memory pools* (`src/lib/mempool.h`) : cette solution est utile pour l'allocation de données permanentes résolvant ainsi le problème avec le *data stack*. Elle consiste en l'allocation d'une réserve de mémoire d'une taille honorable (8 ko par exemple) et ensuite, le programme se charge de redistribuer cette zone en de plus petites entités, voire d'allouer plus de mémoire si nécessaire. Toute la mémoire allouée peut alors être libérée en faisant un `free(3)` sur la réserve. Cette méthode est couramment utilisée par les programmeurs. Ce mécanisme est illustré dans le code suivant :

```
-- src/lib-storage/mailbox-tree.c --

struct mailbox_tree_context *mailbox_tree_init(char separator)
{
        struct mailbox_tree_context *ctx;
        pool_t pool;

        pool = pool_alloconly_create(«mailbox_tree», 10240);

        ctx = p_new(pool, struct mailbox_tree_context, 1);
        ctx->pool = pool;
        ctx->separator = separator;
        return ctx;
}
```

Ne pas garder de données sensibles en mémoire

Une fois que des données sensibles ont été chargées en mémoire et utilisées, il ne sert à rien de les conserver. Prenons l'exemple d'un serveur IMAP. Supposons que lors de l'authentification d'un utilisateur, il garde en mémoire son mot de passe. Un utilisateur malveillant peut alors faire planter le programme en générant un *core*. En fouillant dans le *core*, il récupérera le mot de passe. Il vaut donc mieux effacer les mots de passe une fois qu'ils ont été utilisés. C'est pour cette raison que Dovecot efface les données sensibles présentes en mémoire en utilisant la fonction `safe-memset()` (`src/lib/safe-memset.h`) plutôt que `memset(3)` qui peut être enlevée lors de l'optimisation effectuée par le compilateur [11].

Cependant, il faut aussi se méfier des données qui auraient pu être mises dans le *swap* et donc enregistrées sur le disque dur. La fonction `mlock(2)` est particulièrement recommandée pour éviter la pagination de ce type de données. Son inconvénient principal est que, sur certains systèmes d'exploitation comme Linux, seul l'utilisateur root peut l'invoquer.

C'est pourquoi aucun des programmes présentés dans cet article ne l'utilise (et explique aussi que GnuPG est installé sur certaines distributions avec le `setuid(root)` [14]). Des discussions sont en cours sur la liste de diffusion du noyau Linux pour autoriser l'accès de cette fonction aux utilisateurs [15].

vsftpd

La plupart des choix d'implémentation de vsftpd sont très similaires. Par exemple, les chaînes de caractères sont encapsulées dans une structure appelée `mystr`, et tous les traitements sont réalisés par l'intermédiaire d'une API qui sert de point d'entrée unique.

```
-- ~/vsftpd-2.0.1/str.h --

struct mystr
{
  char* PRIVATE_HANDS_OFF_p_buf;
  unsigned int PRIVATE_HANDS_OFF_len;
  unsigned int PRIVATE_HANDS_OFF_alloc_bytes;
};
```

Les noms des champs n'encouragent pas vraiment l'utilisation directe des données de la structure en court-circuitant l'API :-)

Les fonctions de la libc et les appels systèmes utilisés par vsftpd ont été encapsulés dans des *wrappers* (fichiers `sysutil.c` et `sysdeputil.c`) plus simples à utiliser. Cela permet d'une part d'améliorer la clarté du code, et d'autre part de tester systématiquement les codes d'erreur.

djbdns

Les différents outils constituant djbdns sont dans la même lignée que les autres programmes de D.J. Bernstein : on ne peut pas dire que le code source soit des plus élégants, mais il est d'une efficacité et d'une robustesse à toute épreuve :-)

Toutes les fonctions de traitement de chaînes de caractères ont été ré-implémentées. De manière plus générale, la libc n'est pratiquement pas utilisée, excepté pour la fonction `malloc(3)`.

Postfix

Wietse Venema a lui aussi opté pour le développement de bibliothèques spécifiques pour le traitement des buffers. Pour les chaînes de caractères, il fait appel aux fonctions présentes dans `src/util/vstring.c`. Il a, par exemple, ré-implémenté `strcpy(3)` pour pouvoir recopier des chaînes en toute sécurité :

```
-- src/util/vstring.c --

/*    void    VSTRING_ADDCH(vp, ch)
      VSTRING *vp;
      int     ch;

      void    VSTRING_RESET(vp)
      VSTRING *vp;

      void    VSTRING_TERMINATE(vp)
      VSTRING *vp;

      VSTRING_ADDCH() adds a character to a variable-length string
      and extends the string if it fills up.  vs is a pointer
      to a VSTRING structure; ch the character value to be written.
      The result is the written character.
      Note that VSTRING_ADDCH() is an unsafe macro that evaluates some
      arguments more than once. The result is NOT null-terminated.

      VSTRING_RESET() is a macro that resets the write position of its
      string argument to the very beginning. Note that VSTRING_RESET()
      is an unsafe macro that evaluates some arguments more than once.
      The result is NOT null-terminated.

      VSTRING_TERMINATE() null-terminates its string argument.
      VSTRING_TERMINATE() is an unsafe macro that evaluates some
      arguments more than once.
      VSTRING_TERMINATE() does not return an interesting result.
*/

/* vstring_strcpy - copy string */

VSTRING *vstring_strcpy(VSTRING *vp, const char *src)
{
    VSTRING_RESET(vp);

    while (*src) {
        VSTRING_ADDCH(vp, *src);
        src++;
    }
    VSTRING_TERMINATE(vp);
    return (vp);
}
```

Comme la chaîne peut être augmentée grâce à l'appel à la MACRO `VSTRING_ADDCH()` il n'y a aucun risque de *buffer overflow*.

À la différence de Dovecot, Postfix ne fait pas appel à des *data stacks* ou *memory pool*. Il reste fidèle aux appels `malloc(3)` et `free(3)` mais en les encapsulant dans des fonctions de plus haut niveau : `mymalloc(3)` et `myfree(3)` (`src/util/mymalloc.c`) qui sont notamment utilisées dans la bibliothèque vstring :

```
-- src/util/vstring.c --

/* vstring_alloc - create variable-length string */

VSTRING *vstring_alloc(int len)
{
    VSTRING *vp;

    if (len < 1)
        msg_panic("vstring_alloc: bad length %d", len);
    vp = (VSTRING *) mymalloc(sizeof(*vp));
    vp->vbuf.flags = 0;
    vp->vbuf.len = 0;
    vp->vbuf.data = (unsigned char *) mymalloc(len);
```

```
    vp->vbuf.len = len;
    VSTRING_RESET(vp);
    vp->vbuf.data[0] = 0;
    vp->vbuf.get_ready = vstring_buf_get_ready;
    vp->vbuf.put_ready = vstring_buf_put_ready;
    vp->vbuf.space = vstring_buf_space;
    vp->maxlen = 0;
    return (vp);
}

/* vstring_free - destroy variable-length string */

VSTRING *vstring_free(VSTRING *vp)
{
    if (vp->vbuf.data)
  myfree((char *) vp->vbuf.data);
    myfree((char *) vp);
    return (0);
}
```

Lors de la création d'une chaîne via `vstring_alloc(3)` il faut penser à la libérer après son utilisation :

```
-- src/smtpd/smtpd_check.c --

/* log_whatsup - log as much context as we have */

static void log_whatsup(SMTPD_STATE *state, const char *whatsup,
            const char *text)
{
    VSTRING *buf = vstring_alloc(100);

    vstring_sprintf(buf, «%s: %s: %s from %s: %s;»,
                state->queue_id ? state->queue_id : «NOQUEUE»,
                whatsup, state->where, state->namaddr, text);
    if (state->sender)
        vstring_sprintf_append(buf, « from=<%s>», state->sender);
    if (state->recipient)
        vstring_sprintf_append(buf, « to=<%s>», state->recipient);
    if (state->protocol)
        vstring_sprintf_append(buf, « proto=%s», state->protocol);
    if (state->helo_name)
        vstring_sprintf_append(buf, « helo=<%s>», state->helo_name);
    msg_info(«%s», STR(buf));
    vstring_free(buf);
}
```

La paranoïa est une vertu

Confiance dans les entrées utilisateurs

Bien souvent, le maillon faible de la sécurité des programmes réside dans le traitement (parfois le non-traitement) des données issues du monde utilisateur.

Ces informations peuvent arriver par un grand nombre de canaux de communication : saisie au clavier, variables d'environnement, fichiers, données issues du réseau, etc.

Les quatre programmes pris en exemple ne font aucune confiance aux données « étrangères », et ces dernières sont systématiquement vérifiées et validées avant utilisation. Par exemple, pour les chaînes de caractères, les traitements (effectués dans un contexte non privilégié) passent nécessairement par l'utilisation de bibliothèques spécifiques, au comportement beaucoup plus strict que les fonctions standards, et limitent ainsi grandement les risques d'erreurs.

Confiance en les bibliothèques externes

Les programmes présentés ici ont une très faible confiance en les bibliothèques externes, et encore moins si ces dernières effectuent des traitements très complexes.

On peut également citer toutes les fonctions qui réalisent des opérations dont le programmeur n'a pas forcément conscience.

Ainsi, un appel à `gethostbyaddr(3)` provoque une connexion réseau et nécessite d'analyser la réponse, `getpwnam(3)` va ouvrir un descripteur de fichier sur `/etc/passwd`, etc.

Chris Evans : *« je n'utilise et ne fais confiance qu'à des appels simples de bibliothèques externes, ou alors des appels qui ne manipulent pas les entrées des utilisateurs. Je n'ai aucune confiance en les appels plus compliqués qui manipulent ces entrées. Par exemple, je n'ai pas fait confiance aux fonctions `fnmatch(3)` ou `glob(3)` en partant du fait qu'elles pourraient contenir des failles potentielles de sécurité. Elles ne sont donc pas utilisées dans vsftpd. Et il y a eu des failles dans ces fonctions dans certains systèmes d'exploitation depuis que vsftpd est sorti. »*

Une solution peut être de lire les sources attentivement. C'est d'ailleurs ce que fait **Timo Sirainen** :

« [je n'ai pas confiance en les bibliothèques externes] tout simplement parce qu'elles contiennent du code que je n'ai pas écrit. En général, j'essaie de m'en passer, mais si j'en ai vraiment besoin, je regarde au minimum le code pour l'auditer et comprendre son fonctionnement. En fait, regarder le code plutôt que la documentation fournie avec l'API peut éviter quelques erreurs lors de son utilisation. »

Pour des librairies plus importantes - comme OpenSSL - l'audit n'est pas chose aisée. Il faut alors trouver une autre solution :

Chris Evans : *« à cause de [la] taille [d'OpenSSL] je ne peux pas envisager de développer une alternative. Malheureusement, les utilisateurs de vsftpd ont demandé le support d'OpenSSL. Que faire ? Ma solution a été d'ajouter le support d'OpenSSL [NdT: depuis la version 2.0.0] mais en le désactivant par défaut et en documentant ce choix. Ainsi, c'est aux utilisateurs de prendre la décision de faire confiance ou pas à OpenSSL. »*

Ce choix est d'ailleurs le même dans Postfix :

Wietse Venema : *« pour des raisons pratiques, Postfix utilise la bibliothèque C standard, le noyau du système d'exploitation ainsi que le matériel sous-jacent. Pour les machines de mon domaine porcupine.org, je fais appel aux fournisseurs en lesquels j'ai confiance et j'évite de dépendre de bibliothèques tierces. »*

Dans les sources de Postfix, il est bien précisé que la compilation de Postfix avec le support SASL rendra le programme aussi sûr que les autres MTA compilés avec ce support :

```
-- README_FILES/SASL_README --
«People who go to the trouble of installing Postfix may have the expectation that
Postfix is more secure than some other mailers. The Cyrus SASL library is a lot
of code. With SASL authentication enabled in the Postfix SMTP client and SMTP
server, Postfix becomes as secure as other mail systems that use the Cyrus SASL
library.»
```

Le choix est tout autre pour **Timo Sirainen** :

> *« Si la librairie est constituée de beaucoup de sources et qu'il n'y a aucun moyen pour que je sois certain qu'elle est sûre, je vais essayer de minimiser l'impact que pourrait avoir un trou de sécurité potentiel. Par exemple, tout le code SSL de Dovecot est appelé dans des processus isolés, non privilégiés et chrootés. Même si un attaquant arrive à exploiter la bibliothèque SSL, il ne pourra pas lire les mails des autres utilisateurs. »*

Un autre cas se présente lors de l'appel à un programme externe à qui l'on sous-traite une tâche particulière. Notre programme a beau être sûr, audité, vérifié, il dépend quand même d'un autre programme sur lequel nous n'avons aucun contrôle. Qu'arrivera-t-il si ce programme présente une faille ? Serons-nous vulnérables ? La sécurité de notre programme est alors celle de cette commande externe.

Dans les programmes qui nous intéressent, c'est le cas de certains serveurs FTP (wu-ftpd pour ne pas le citer) : pour afficher le contenu d'un répertoire, ils se basent sur `/bin/ls` autorisant ainsi l'utilisateur à fournir des arguments à cette commande. La sécurité du serveur est dépendante de celle de `/bin/ls` qui fera peut-être appel à `fnmatch(3)` ou `glob(3)`, fonctions dans lesquelles nous ne pouvons avoir confiance.

Chris Evans a donc recodé un appel qui simule le fonctionnement de la commande `/bin/ls`. Par effet de bord, il s'est rendu compte que cela a permis d'augmenter les performances (pas de `fork(3)` ni d'`exec(3)`).

Validation du code source, audit, réflexions sur le développement

Validation du source

Il est facile d'écrire du code et encore plus simple d'introduire des vulnérabilités à cause d'une erreur de programmation. Qui n'a jamais oublié un `i++` dans son code à la fin d'une dure journée (nuit ?) de programmation intensive ?

Pour empêcher de telles erreurs, on peut utiliser des programmes facilitant la découverte de ces erreurs ou qui vérifient la gestion de la mémoire comme Valgrind [12].

Nous avons demandé aux différents auteurs s'ils utilisent de tels programmes et les réponses sont partagées :

> **Chris Evans** : *« je n'utilise pas de vérificateur de code. Je n'ai pas encore trouvé d'analyseur de source statique qui serait suffisamment puissant pour m'être utile dans la recherche de failles de sécurité. (Mon prochain projet important pourrait être un puissant chercheur de vulnérabilité mi-statique, mi-dynamique). Je n'ai jamais utilisé Valgrind non plus : vsftpd n'a généralement pas de problème de fuite de mémoire à cause de la philosophie du « faire simple » (KISS) pour tout ce qui est de la gestion de la mémoire et des buffers. »*

> **Timo Sirainen** : *« la plupart des vérificateurs de code source prévu pour « un audit rapide du code » semblent être inutiles. Ils servent juste à souligner les endroits où sont appelées des fonctions définies par ces programmes comme « non sûres ». »*

D'autres vérificateurs sont beaucoup plus intéressants et utiles. Ainsi, le *Stanford Checker* [NdT: **http://metacomp.stanford.edu/**], qui analyse de manière statique le code et qui peut signaler des bogues peu évidents, a trouvé de nombreux bogues dans le noyau Linux. Il est dommage que de tels outils ne soient généralement pas disponibles pour le commun des mortels (la dernière fois que j'ai regardé).

Valgrind est assez utile pour identifier tous les problèmes liés à l'utilisation de la mémoire. Il est préférable de ne jamais ressentir le besoin de l'utiliser, mais si c'est le cas, c'est un outil très précieux. Je ne l'utilise plus que très rarement, souvent pour savoir pourquoi mon nouveau code ne fonctionne pas comme prévu et il peut me montrer que j'ai oublié d'initialiser certaines variables.

> **Wietse Venema** : *« les tests [avant chaque sortie d'une version de Postfix] incluent le fonctionnement du système sous l'œil d'un débogueur de mémoire et d'un programme de vérification des fuites de mémoire. Je ne conçois pas de mettre à disposition une nouvelle version sans utiliser de tels programmes. »*

Audit

Même les programmeurs les plus compétents ont besoin que leur code soit relu par des personnes tierces pour que ces dernières puissent donner un avis externe sur le source, voire détecter des bogues. Mais, pour paraphraser Timo Sirainen, auditer des logiciels non sécurisés ne les rend pas plus sécurisés. Sendmail est, là encore, un excellent exemple : il a été audité un grand nombre de fois par des personnes compétentes et des vulnérabilités sont apparues malgré cela.

Tout au long de cet article, nous parlons de l'audit du code source écrit. Mais qu'en est-il vraiment ? Est-ce que le fait qu'un logiciel soit « open source » simplifie ou augmente l'audit du code ? Laissons les programmeurs répondre :

> **Chris Evans** : *« je crois que d'autres personnes ont vérifié le code de vsftpd. Cependant, je ne pense pas qu'elles soient nombreuses. Mais au moins une personne très compétente l'a regardé : le développeur principal d'OpenWall Linux* [NdT : Solar Designer - **http://www.openwall.com/**]. *Pour chaque nouvelle version, je ne développe que des ajouts ou des extensions très simples. En gardant ces ajouts simples, les chances d'introduire des bogues de sécurité sont très faibles. Je crois que tout le code complexe de vsftpd est déjà réalisé (par exemple le code pour le traitement des chaînes de caractères, celui pour lister le contenu des répertoires, etc.) »*

> **Timo Sirainen** : *« j'aimerais que des gens auditent le code de Dovecot et me disent quels sont les problèmes potentiels. La difficulté est que peu de personnes le font gratuitement et qu'il est donc difficile d'en trouver. Cependant, j'envisage d'offrir une prime à la première personne qui trouvera un trou de sécurité dans la version 1.0 de Dovecot. J'espère que ça encouragera plus de personnes à l'auditer et qu'elles concluront que le code est sûr :)*
>
> *Pour ma part, j'ai plusieurs fois effectué un audit rapide du code depuis sa naissance. En outre, toutes mes entrées CVS sont envoyées à la liste dovecot-cvs que je relis pour chaque « commit ». En les lisant, je remarque de temps en temps des erreurs.*
>
> *Avant que Dovecot atteigne la version 1.0, je prévois d'auditer intensivement tout le code. »*

Wietse Venema : « *avec chaque nouvelle version de Postfix, les différences ligne à ligne du code source sont vérifiées manuellement et testées en faisant réellement fonctionner le programme. Tous les embranchements if/else sont testés aux endroits nécessaires. De plus, on enregistre le processus de vérification en l'écrivant à l'encre rouge sur les sorties papier des différences ligne à ligne.* »

Ces traces sont aussi conservées pour vsftpd : le fichier AUDIT présent dans les sources résume le niveau d'audit de chaque fichier C. Une note de 1 (pas audité) à 5 (audité intensément par plusieurs personnes compétentes) est affectée. Une règle intéressante est le fait que dès qu'un fichier change, sa note retombe à 1, à moins que les changements aient été audités très attentivement eux aussi. Chacun peut ainsi se faire une idée du niveau de relecture du code. Pour indication, la plupart des notes sont 2 ou 3. Malheureusement, ce fichier n'a pas l'air d'avoir été mis à jour depuis septembre 2003...

Réflexions sur le développement

Si l'on regarde le processus de développement des quatre programmes présentés, on se rend vite compte que très peu de personnes sont responsables du code. À l'exception de Postfix, un seul développeur est impliqué pour chacun.

Peut-on en déduire qu'un programme sécurisé doit être développé par une personne unique ? Nous avons posé la question aux intéressés :

Chris Evans : « *Je pense qu'un projet impliquant un nombre important de personnes peut être sécurisé à condition que tous les développeurs aient une connaissance aiguë de la sécurité, ou qu'une telle personne supervise tous les changements. Dans le cas de vsftpd, je suis le seul développeur.* »

Timo Sirainen : « *Je pense qu'un programme qui se proclame sécurisé devrait avoir une seule personne responsable pour tous les trous de sécurité présents. Ou, si le programme est très volumineux, il peut être divisé en plusieurs petites parties ayant chacune un tel responsable. Cette personne doit regarder tout le code et s'assurer qu'il est sûr.*

S'excuser en disant « désolé, mais ce trou de sécurité a été causé par un patch fait par quelqu'un d'autre et je ne suis donc pas responsable » fait penser que le processus de développement n'est pas effectué correctement.

La contrepartie d'avoir une seule personne qui réalise tout l'audit est bien entendu que le délai d'inclusion de patches importants peut être long avant qu'ils soient audités et acceptés. Mais je ne pense pas que l'inclusion rapide d'un code développé par un tiers soit une excuse pour introduire un trou de sécurité dû à un audit peu attentionné. »

Wietse Venema : « *À l'heure actuelle, je maintiens le contrôle éditorial [de Postfix]. Le nombre de personnes qui peuvent soumettre du code source Postfix exempt de tout problème est très restreint (trois).*

Un domaine connexe est l'inclusion des patches dans un programme. Comment se déroule la validation d'un patch ? La réponse de **Chris Evans** est très claire :

Quand je reçois un patch, je l'applique toujours manuellement pour mieux comprendre l'impact des changements et être sûr qu'il n'y a pas de modifications subtiles concernant la sécurité. Cela étant, je ne reçois pas beaucoup de patches. »

Conclusion

L'objectif de cet article n'était pas de couvrir l'intégralité des concepts et des techniques utilisés en programmation sécurisée, mais plutôt d'illustrer quelques grands principes de base à l'aide de programmes connus (et reconnus).

Le lecteur souhaitant approfondir un sujet particulier pourra se référer à l'abondante documentation disponible en ligne. Cependant, l'étude soigneuse du code source de programmes réputés pour leur sécurité reste le moyen le plus formateur. Pour ceux qui prennent l'article en cours de route, si vous ne deviez retenir qu'une chose : KISS ! :-)

Références

[1] *Secure Programming for Linux and Unix HOWTO* **http://www.dwheeler.com/secure-programs/**

[2] vsftpd, **http://vsftpd.beasts.org/**

[3] djbdns, **http://cr.yp.to/djbdns.html**

[4] Postfix, **http://www.postfix.org/**

[5] Dovecot, **http://dovecot.org/**

[6] *Linux Capabilities* **http://www.kernel.org/pub/linux/libs/security/linux-privs/**

[7] Timo Sirainen - *Secure Software* **http://irccrew.org/~cras/security/securesoftware.html**

[8] Timo Sirainen - *Dovecot Design*, Fichier `doc/design.txt` dans les sources de Dovecot

[9] Timo Sirainen - *Secure, Efficient and Easy C programming* **http://irccrew.org/~cras/security/c-guide.html**

[10] Timo Sirainen - *Secure Coding* **http://dovecot.org/doc/securecoding.txt** Fichier `doc/securecoding.txt` dans les sources de Dovecot

[11] *Software leaves encryption keys, passwords lying around in memory,* **http://online.securityfocus.com/archive/82/298061/2002-10-28/2002-11-03/1**

[12] Valgrind - A GPL'd system for debugging and profiling x86-Linux programs, **http://valgrind.kde.org**

[13] Bradley Spengler - *chroot(), sécurité illusoire ou illusion de sécurité ?* MISC 16 **http://www.miscmag.com/articles/index.php3?page=1008**

[14] *Why do I get «gpg: Warning: using insecure memory!» ?* **http://www.gnupg.org/documentation/faqs.html#q6.1**

[15] Allowing non-root user to mlock memory **http://www.kerneltraffic.org/kernel-traffic/kt20040906_273.html#5**

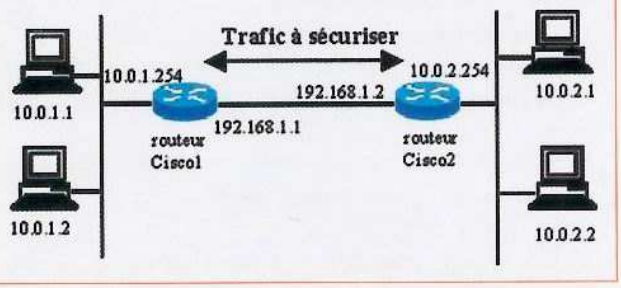
[16] Timo Sirainen - *Most common security flaws with C programs* **http://irccrew.org/~cras/security/flaws.html**

[17] Chris Evans - *The importance of trust and trust relationships* **http://vsftpd.beasts.org/TRUST**

IPSEC entre deux routeurs CISCO

Cette fiche technique décrit les différentes phases de configuration d'un routeur Cisco pour la mise en place d'un tunnel IPSEC entre deux réseaux locaux via deux routeurs Cisco. Elle s'appuie sur l'article « IPSEC et routeur Cisco » du MISC 15 [1], où il avait été question d'installer IPSEC sur un routeur Cisco qui était client d'une passerelle sous OpenBSD. L'authentification mutuelle des deux extrémités du tunnel avait été réalisée en mode manuel de gestion des clés, puis en mode automatique avec un secret pré-partagé. Cette fois-ci, on ne décrira que le mode de gestion automatique des clés basé sur le protocole ISAKMP, la configuration manuelle n'étant pas le mode le plus utilisé, IPSEC sera paramétré dans un premier temps avec un secret pré-partagé, puis dans un deuxième temps avec le système de certificats X509.

Figure 1



Utilisation de l'IOS de Cisco avec le protocole ISAKMP

1- Utilisation de l'IOS de Cisco avec un secret pré-partagé

Les configurations des deux routeurs sont symétriques l'une par rapport à l'autre. Il suffit donc de reprendre ce qui avait été détaillé dans l'article [1] et de le transcrire parallèlement sur le routeur distant.

On commence par configurer les paramètres pour la politique de sécurité ISAKMP sur les deux routeurs.

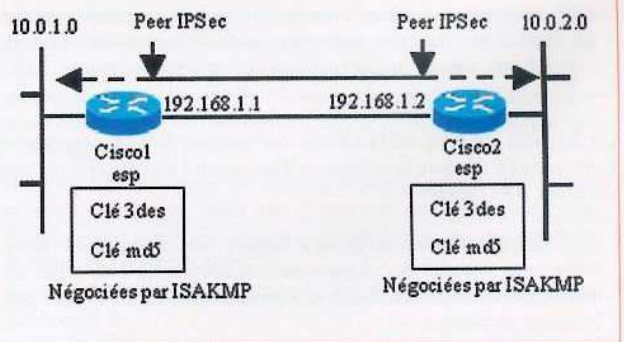
```
Cisco1(config)#crypto isakmp policy 10
Cisco1(config-isakmp)#encr 3des
Cisco1(config-isakmp)#hash md5
Cisco1(config-isakmp)#authentication pre-share
Cisco1(config-isakmp)#group 2
Cisco1(config-isakmp)#exit

Cisco1(config)#crypto isakmp key mekmitasdigoat address 192.168.1.2

Cisco2(config)#crypto isakmp policy 10
Cisco2(config-isakmp)#encr 3des
Cisco2(config-isakmp)#hash md5
Cisco2(config-isakmp)#authentication pre-share
Cisco2(config-isakmp)#group 2
Cisco2(config-isakmp)#exit

Cisco2(config)#crypto isakmp key mekmitasdigoat address 192.168.1.1
```

La négociation de phase 1 étant faite, on configure les paramètres de négociation pour la phase 2. On définit la `transform-set`, la `crypto map` et l'`access-list` qui décrit le trafic. On remarque ici que le trafic à sécuriser correspond aux données échangées entre les deux sous-réseaux 10.0.1.0 et 10.0.2.0, les hôtes IPSec (`peer`) restant les deux interfaces FastEthernet des Cisco en 192.168.1.1 et 192.168.1.2.

```
Cisco1(config)#crypto ipsec transform-set chiff esp-3des esp-md5-hmac

Cisco1(config)#crypto map cisco1-cisco2 10 ipsec-isakmp
Cisco1(config-crypto-map)#set peer 192.168.1.2
Cisco1(config-crypto-map)#set transform-set chiff
Cisco1(config-crypto-map)#match address 110

Cisco1(config)#access-list 110 permit ip 10.0.1.0 0.0.0.255 10.0.2.0 0.0.0.255 log

Cisco1(config)#interface FastEthernet0
Cisco1(config-if)#ip address 192.168.1.1 255.255.255.0
Cisco1(config-if)#crypto map cisco1-cisco2


Cisco2(config)#crypto ipsec transform-set chiff esp-3des esp-md5-hmac

Cisco2(config)#crypto map cisco1-cisco2 10 ipsec-isakmp
Cisco2(config-crypto-map)#set peer 192.168.1.1
Cisco2(config-crypto-map)#set transform-set chiff
Cisco2(config-crypto-map)#match address 110

Cisco2(config)#access-list 110 permit ip 10.0.2.0 0.0.0.255 10.0.1.0 0.0.0.255 log

Cisco2(config)#interface FastEthernet0/0
Cisco2(config-if)#ip address 192.168.1.2 255.255.255.0
Cisco2(config-if)#crypto map cisco1-cisco2
```

Le tunnel IPSec est ainsi mis en place entre les deux réseaux 10.0.2.0 et 10.0.1.0. On peut vérifier les configurations avec les commandes citées dans l'article [1] et visualiser de la même manière que dans le premier cas les SA (*Security Association*) établies.

Figure 2



Marie Bordesoules
mary@rstack.org
Ingénieur-Chercheur au Commissariat à L'Energie Atomique

Cette méthode a un inconvénient : si on souhaite augmenter le nombre de routeurs Cisco capables de dialoguer en IPSec, il faut ajouter sur chaque nouveau routeur la clé commune pré-partagée, clé qui figure en clair dans le fichier de configuration des routeurs. La méthode d'authentification mutuelle à l'aide des certificats résout ce problème.

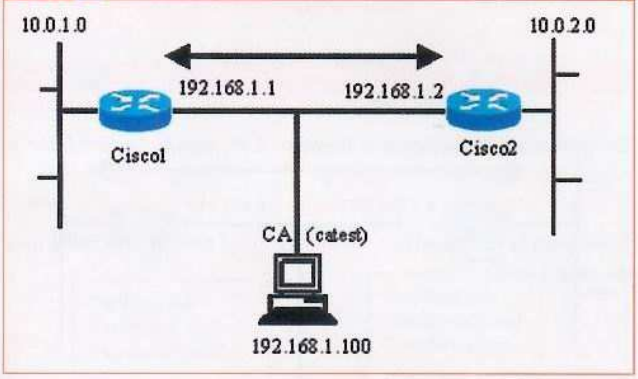
2- Utilisation de l'IOS de Cisco avec les certificats X509

L'IOS de Cisco supporte la gestion des certificats X509.

Chaque extrémité du tunnel possède un certificat certifié (ou signé) par une autorité de certification. L'authentification mutuelle sur laquelle se base la phase 1 de la mise en place du tunnel est ici gérée par les certificats.

Il faut commencer par avoir une autorité de certification (CA) capable de recevoir des demandes de certificats venant des routeurs Cisco. Ici, nous avons utilisé un serveur Windows 2000 intégrant CEP (*Certificate Enrollment Protocol*) pour l'enregistrement des routeurs Cisco sur une autorité de certification (installation de mscep.dll).

Figure 3



Sur chaque extrémité, on doit mettre à jour l'heure et la date (`clock timezone` et `clock set`) qui sont des facteurs indispensables pour l'intégration dans un système de certificats.

```
ciscol(config)#clock timezone cest +1
ciscol#clock set 17:47:21 21 june 2004
ciscol#sh clock
17:47:25.011 cest Mon Jun 21 2004
```

Il est important aussi de donner un nom de machine ainsi qu'un nom de domaine aux routeurs :

```
ciscol(config)#ip domain-name misc.com
```

Ensuite, il faut définir sur chaque routeur l'autorité de certification, c'est-à-dire entrer son nom catest ainsi que son adresse IP (`ip host`), et spécifier ses caractéristiques (`crypto ca trustpoint`), c'est-à-dire son mode d'enregistrement (`ra`), sa localisation (`http://catest`) et le protocole utilisé (`CEP`) :

```
ciscol(config)#ip host catest 192.168.1.100

ciscol(config)#crypto ca trustpoint catest
ciscol(ca-trustpoint)#enrollment mode ra
ciscol(ca-trustpoint)#enrollment url http://catest/certsrv/mscep/mscep.dll
ciscol(ca-trustpoint)#exit
On peut vérifier dans le fichier de configuration que l'autorité est bien
déclarée (#sh conf) :
crypto ca trustpoint catest
 enrollment mode ra
 enrollment url http://catest:80/certsrv/mscep/mscep.dll
!
```

Puis chaque routeur génère son propre couple de clés (publique/privé) par la commande `crypto key generate rsa`. Soit on ne génère qu'un seul couple de clés pour l'authentification et le chiffrement (`general-keys`) soit un couple pour chaque utilisation (`usage-keys`).

```
ciscol(config)#crypto key generate rsa usage-keys
```

On peut visualiser les clés ainsi générées par la commande `#sh crypto key mypubkey rsa` :

```
ciscol#sh crypto key mypubkey rsa
% Key pair was generated at: 17:48:14 cest Jun 21 2004
Key name: ciscol.misc.com
Usage: Signature Key
Key is not exportable.
Key Data:
 305C300D 06092A86 4886F70D 01010105 0003480D 30480241 00CF3E95 A33965E0
 A05A4B4B F7CC91DD 7E8D7A93 97C5827F F7A9EB6A 716A15C4 DB85C0B2 25C99A2D
 436B9984 1397787A FF6F4D1D B0EF60C2 7CCCD9C6 82B8AC6D DF020301 0001
% Key pair was generated at: 17:48:17 cest Jun 21 2004
Key name: ciscol.misc.com
Usage: Encryption Key
Key is not exportable.
Key Data:
 305C300D 06092A86 4886F70D 01010105 0003480D 30480241 00C94CEE 28ED2A57
 7B99B4FB 831869C3 6894709E AE018037 3969EE78 1B34FEF1 A0869214 76715F84
 42DB279A 3172C05C 5F8AC284 92637B8D 4B6A8CF5 D70F0DF3 8B020301 0001
% Key pair was generated at: 17:48:22 cest Jun 21 2004
Key name: ciscol.misc.com.server
Usage: Encryption Key
Key is exportable.
```

Avant que les routeurs n'envoient une demande de certificat à l'autorité catest, ils doivent récupérer le certificat de celle-ci par la commande `crypto ca authenticate` :

```
ciscol(config)#crypto ca authenticate catest
```

La communication entre l'autorité et le routeur se met immédiatement en place. Le routeur doit alors vérifier l'identité du CA avant de faire de catest une autorité de confiance. L'autorité envoie donc son certificat auto-signé (`fingerprint`) et l'utilisateur compare manuellement la signature envoyée (`fingerprint`) avec des données échangées préalablement en contactant l'administrateur du CA :

```
Certificate has the following attributes:
Fingerprint: B07B18AD 9F69F89E 138819E4 C159CD91
% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.
ciscol(config)#exit
```

Le certificat de l'autorité est alors installé sur le routeur. On peut aller le visualiser dans le fichier de configuration (#sh conf) :

```
!
crypto ca certificate chain catest
 certificate ca 05EA0202A247FAB542B8632EB8D11E4C nvram:LaCAdemoutan#1E4CCA.cer
```

Enfin, chaque hôte envoie sa demande de certificat pour le couple de clés qu'il a généré précédemment (crypto ca enroll). Lors de l'enregistrement de l'hôte sur l'autorité de certification, le CA doit vérifier l'identité du client par la demande d'un mot de passe (%Create a challenge password).

```
ciscol(config)#crypto ca enroll catest
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this
   password to the CA Administrator in order to revoke your certificate.
   For security reasons your password will not be saved in the configuration.
   Please make a note of it.

Password:
Re-enter password:

% The fully-qualified domain name in the certificate will be: ciscol.misc.com
% The subject name in the certificate will be: ciscol.misc.com
% Include the router serial number in the subject name? [yes/no]: no
% Include an IP address in the subject name? [no]: no
Request certificate from CA? [yes/no]: yes
% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.
ciscol(config)#
    Signing Certificate Request Fingerprint:
    E276150D BC4ED5B7 7742508C C3158214
    Encryption Certificate Request Fingerprint:
    F6FD3256 281C728A 9015B3EF A140347D
```

Grâce au mode debug (#debug crypto pki transactions, #debug crypto pki messages), on peut suivre l'évolution de la demande de certificat auprès de l'autorité (statut d'attente, puis certificat reçu par le routeur) :

```
Jun 21 16:58:41.795: CRYPTO_PKI: status = 102: certificate request pending
Jun 21 16:59:03.679: CRYPTO_PKI: status = 102: certificate request pending
Jun 21 16:59:04.379: CRYPTO_PKI: status = 102: certificate request pending
Jun 21 16:59:26.263: CRYPTO_PKI: status = 102: certificate request pending
Jun 21 17:00:05.843: %CRYPTO-6-CERTRET: Certificate received from Certificate
Authority
ciscol(config)#
Jun 21 17:00:28.419: %CRYPTO-6-CERTRET: Certificate received from Certificate
Authority
```

À ce stade, les certificats du routeur (il y en a deux si on a généré deux couples de clés) ainsi que le certificat de la CA sont installés sur la machine. On peut les visualiser en tapant #sh crypto ca certificates :

```
ciscol#sh crypto ca certificates
Certificate
Status: Available
Certificate Serial Number: 49AFAA7300000000000006
Certificate Usage: Encryption
Issuer:
CN = La CA de moutane
Subject:
Name: ciscol.misc.com
OID.1.2.840.113549.1.9.2 = ciscol.misc.com
CRL Distribution Point:
http://catest/CertEnroll/La%20CA%20de%20moutane.crl
Validity Date:
start date: 16:56:43 cest Jun 21 2004
end date: 17:06:43 cest Jun 21 2005
renew date: 01:00:00 cest Jan 1 1970
Associated Trustpoints: catest

Certificate
Status: Available
Certificate Serial Number: 49AF906300000000000005
Certificate Usage: Signature
Issuer:
CN = La CA de moutane
Subject:
Name: ciscol.misc.com
OID.1.2.840.113549.1.9.2 = ciscol.misc.com
CRL Distribution Point:
http://catest/CertEnroll/La%20CA%20de%20moutane.crl
Validity Date:
start date: 16:56:36 cest Jun 21 2004
end date: 17:06:36 cest Jun 21 2005
renew date: 01:00:00 cest Jan 1 1970
Associated Trustpoints: catest

CA Certificate
Status: Available
Certificate Serial Number: 05EA0202A247FAB54288632EB8D11E4C
Certificate Usage: Signature
Issuer:
CN = La CA de moutane
Subject:
CN = La CA de moutane
CRL Distribution Point:
http://catest/CertEnroll/La%20CA%20de%20moutane.crl
Validity Date:
start date: 15:24:16 cest May 13 2004
end date: 15:33:20 cest May 13 2009
Associated Trustpoints: catest
```

On peut aussi vérifier la chaîne de certification dans le fichier de configuration (#sh conf) : le certificat de l'autorité est enregistré et les certificats de l'hôte local sont certifiés par l'autorité de confiance nommée catest.

```
crypto ca certificate chain catest
 certificate 49AFAA7300000000000006 nvram:LaCAdemoutan#6.cer
 certificate 49AF906300000000000005 nvram:LaCAdemoutan#5.cer
 certificate ca 05EA0202A247FAB54288632EB8D11E4C nvram:LaCAdemoutan#1E4CCA.cer
```

On peut aussi télécharger la dernière CRL depuis l'autorité sur le routeur (crypto ca crl request) :

```
ciscol(config)#crypto ca crl request catest
```

Et on peut la visualiser (#sh crypto ca crls) avec sa prochaine date de mise à jour :

```
ciscol#sh crypto ca crls
CRL Issuer Name:
CN = La CA de moutane
LastUpdate: 09:33:01 cest Jun 21 2004
NextUpdate: 21:53:01 cest Jun 28 2004
Retrieved from CRL Distribution Point:
http://catest/CertEnroll/La%20CA%20de%20moutane.crl
```

Lorsque le routeur veut identifier un certificat d'un hôte distant, il regarde si ce certificat n'est pas dans une liste de révocation (CRL). Si le routeur n'a pas de liste de révocation et n'est pas capable d'en obtenir une, il peut accepter quand même le certificat de l'hôte distant à condition de mettre l'option crl optional :

```
ciscol(config)#crypto ca trustpoint catest
ciscol(ca-trustpoint)#crl optional
```

La configuration pour le routeur distant est complètement symétrique. À ce stade, l'authentification mutuelle sur laquelle se base la politique de sécurité de phase 1 du protocole ISAKMP/IPSEC est mise en place. Il suffit donc de paramétrer les politiques de sécurité de phase 1 (crypto isakmp policy 1) en spécifiant qu'on utilise le mode certificat (authentication rsa-sig) puis de phase 2

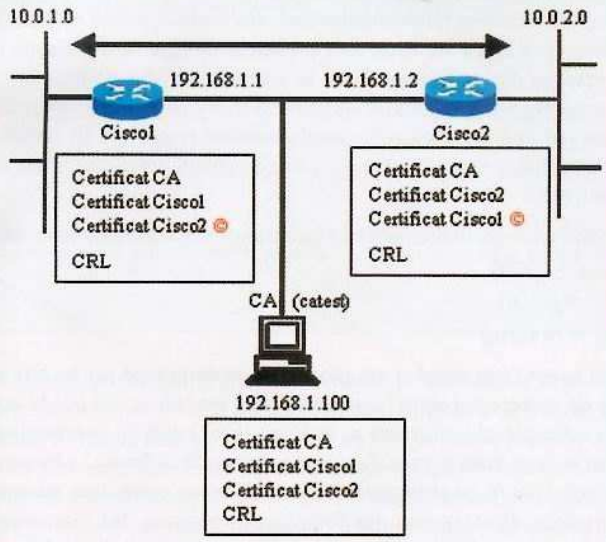
(`crypto ipsec transform-set`, `crypto map`, `access-list`) aux deux extrémités comme dans les cas précédents :

```
ciscol(config)#crypto isakmp policy 1
ciscol(config-isakmp)#encr 3des
ciscol(config-isakmp)#hash md5
ciscol(config-isakmp)#auth
ciscol(config-isakmp)#authentication rsa-sig
ciscol(config-isakmp)#group 2
ciscol(config-isakmp)#exit
ciscol(config)#crypto ipsec transform-set chiff esp-3des esp-md5-hmac
ciscol(cfg-crypto-trans)#crypto map ciscol-cisco2 10 ipsec-isakmp
% NOTE: This new crypto map will remain disabled until a peer
and a valid access list have been configured.
ciscol(config-crypto-map)# set peer 192.168.1.2
ciscol(config-crypto-map)#set transform-set chiff
ciscol(config-crypto-map)# match address 110
ciscol(config-crypto-map)#exit
ciscol(config)#access-list 110 permit ip host 192.168.1.1 host 192.168.1.2
ciscol(config)#int fastEthernet 0
ciscol(config-if)#crypto map ciscol-cisco2
ciscol(config-if)#end
```

Pour établir une communication sécurisée entre les deux hôtes, ciscol envoie un ping à cisco2. Lors de la phase 1, les certificats sont échangés entre les deux hôtes puis vérifiés sur chaque routeur. La commande `#sh crypto key pubkey-chain rsa`, lancée sur le routeur cisco2, permet de visualiser les clés des hôtes distants certifiées manuellement (`M`) et les clés des hôtes distants certifiées par le système de certification (`C`), ici celles du routeur ciscol.

```
cisco2#sh crypto key pubkey-chain rsa
Codes: M - Manually configured, C - Extracted from certificate

Code    Usage   IP-Address/VRF      Keyring          Name
C       Signing                             default          X.500 DN name:
                    CN = La CA de moutane
```

Figure 4



En cas de souci, les commandes de débogue utiles pour le mode certificats X509, sont :

```
#debug crypto key-exchange
#debug crypto pki messages
#debug crypto pki transactions
```

Un peu de filtrage

Dans les trois cas (mode manuel de gestion des clés vu dans l'article [1], secret pré-partagé et certificats X509), pour sécuriser un peu plus notre système, il est possible de filtrer uniquement le trafic IPSec aux entrées du tunnel. Pour cela, il suffit d'ajouter des règles de filtrage en entrée des interfaces des deux extrémités du tunnel, le protocole de gestion des clés ISAKMP utilisant le port 500 du protocole UDP.

On crée alors un `access-group` qui regroupe plusieurs règles de filtrage. La dernière règle n'est pas obligatoire, car les deux précédentes signifient déjà que tout le reste du trafic est ignoré, mais elle est utile pour déboguer (`log`) :

```
Ciscol(config)#interface FastEthernet0
Ciscol(config-if)#ip address 192.168.1.1 255.255.255.0
Ciscol(config-if)#ip access-group 120 in
Ciscol(config-if)#crypto map ciscol-cisco2

Ciscol(config)#access-list 120 permit udp host 192.168.1.2 eq isakmp host
192.168.1.1 eq isakmp log
Ciscol(config)#access-list 120 permit esp host 192.168.1.2 host 192.168.1.1 log
Ciscol(config)#access-list 120 deny   ip any any log
```

Ainsi, sur le routeur ciscol, seul le trafic IPSEC venant de l'hôte cisco2 est autorisé à entrer sur l'interface FastEthernet0 du ciscol.

Conclusion

Comme dans le cas de l'article [1], l'objectif de cette fiche pratique n'est pas de décrire de façon très détaillée la mise en place d'IPSec avec toutes ses configurations possibles, mais de donner les premières bases techniques pour utiliser IPSec sur un routeur Cisco. Cet article montre que le protocole IPSec peut être installé sur des routeurs Cisco pour sécuriser du trafic. Dans le cas d'un petit nombre de réseaux à interconnecter, la méthode avec secret pré-partagé peut être suffisante. Cependant, si les routeurs Cisco à configurer sont en quantité importante et variable dans le temps, la méthode des certificats X509 apparaît comme la plus adaptée.

Merci à Mathieu BLANC (**moutane@rstack.org**) et à Laurent OUDOT (**oudot@rstack.org**) ainsi qu'à mes collègues du laboratoire Conception Développement et Système.

Références

[1] Fiche technique *« IPSec et routeur Cisco »*, MISC 15
[2] **http://www.cisco.com/**
[3] **http://www.cisco.com/univercd/home/home.htm**
[4] **http://www.cisco.com/univercd/cc/td/doc/pcat/index.htm**
[5] **http://www.cisco.com/warp/public/707/nle_toc.html**
[6] Dossier IPSec *« Sécurisation de communications WIFI avec IPSec »*, MISC 10
[7] Fiche technique *« IPSec sur OPenBSD, FreeBSD et Mac OS X »*, dans MISC 10
[8] Fiche technique *« IPSec et Linux »*, dans MISC 10

Le chiffrement par flot

Depuis la sélection de l'algorithme Rijndaël, en octobre 1999, par le gouvernement américain, pour remplacer le Data Encryption Standard (D.E.S) par l'Advanced Encryption Standard (A.E.S.), algorithme de chiffrement par blocs, une nette tendance se dessine dans une partie de la communauté internationale de cryptologie au profit des technologies par blocs. Cet article a pour but d'introduire le chiffrement par flot qui demeure, de manière quasi-exclusive, employé pour les usages gouvernementaux de la plupart des pays. Ce chiffrement présente de très nombreuses qualités qui n'ont pas été remises en question, ni scientifiquement ni opérationnellement. Il existe plusieurs catégories de systèmes et leur conception, lorsqu'un haut niveau de sécurité est requis, n'est pas seulement une question de théorie mathématique mais réclame surtout un véritable art de l'ingénieur. C'est la raison pour laquelle ces systèmes sont le plus souvent propriétaires et les algorithmes non publics.

Introduction : la théorie

Le terme de chiffrement par flot (*stream cipher*) désigne un chiffrement « à la volée », c'est-à-dire que les caractères d'un message clair sont chiffrés au fur et à mesure qu'ils sont produits. Cela explique pourquoi ces systèmes (lorsqu'ils sont bien implémentés) sont les champions toutes catégories en termes de vitesse de chiffrement. C'est l'une des raisons pour laquelle ils sont prioritairement utilisés dans les environnements embarqués (satellites), mobiles (téléphonie) ou sans fil (**Wep**, *Bluetooth*).

L'autre raison d'une utilisation quasi-exclusive pour les moyens de chiffrement gouvernementaux tient dans le fait que ces systèmes, lorsqu'ils sont bien conçus, offrent un très haut niveau de sécurité et, qui plus est, la théorie, et surtout l'art de l'ingénieur, permettent de s'assurer que ce niveau est bien atteint et réel.

Le corpus de connaissances théoriques sur les primitives, c'est-à-dire les briques mathématiques de base utilisées pour construire de tels systèmes (registres, fonctions booléennes...), est très important. Les quelques attaques connues n'envisagent ces primitives que dans un contexte simple d'utilisation, autrement dit quand ces primitives sont utilisées telles quelles pour réaliser le système lui-même. La réalité est toute autre et c'est là que l'art de l'ingénieur cryptologue intervient d'une manière essentielle et incontournable. Ces différentes primitives, que nous exposerons un peu plus loin, sont en fait combinées entre elles, de manière à gommer leurs faiblesses respectives pour produire au final un système le plus simple possible (nécessité pour la phase industrielle d'implémentation, car ces systèmes sont le plus souvent implémentés dans des puces spécifiques appelées cryptoprocesseurs) et s'approchant de manière optimale d'un système aléatoire parfait. Une phase longue et intensive de tests statistiques validera au final la qualité du système.

Les systèmes par flot sont des systèmes symétriques (encore appelés à clé secrète). Cela signifie qu'émetteur et destinataire se partagent une même clé secrète qui doit donc être mise en place avant toute communication.

Ces systèmes sont de deux types : synchrones (les plus courants) ou asynchrones.

Les systèmes synchrones

Pour ces systèmes, le chiffrement s'effectue par combinaison (le plus souvent une addition *modulo* 2 ou `xor`) d'une suite aléatoire vraie (comme dans le système dit de *Vernam* ; voir plus loin) ou d'une suite dite pseudo-aléatoire (ou encore suite PN pour *Pseudo-Noise* : littéralement assimilable à du bruit ; le terme de *pseudo-aléatoire* sera expliqué plus loin) et du texte clair pour produire un texte chiffré ou cryptogramme.

Définition
Un système de chiffrement par flot synchrone est un système de chiffrement dans lequel la suite aléatoire vraie ou pseudo-aléatoire est générée indépendamment du message clair et du message chiffré.

Dans ce qui suit, nous considèrerons uniquement les systèmes pseudo-aléatoires, qui, en pratique, à l'exception de quelques communications très sensibles, sont les seuls systèmes utilisés. La différence entre les deux sera explicitée lorsque sera présenté le système dit de *Vernam*. Pour le moment, et afin de faciliter la compréhension, voyons un système de chiffrement pseudo-aléatoire comme une approximation satisfaisante en pratique d'un système mathématiquement parfait mais extrêmement difficile à mettre en œuvre.

L'opération de chiffrement se résume alors aux équations suivantes :

$s_{i+1} = f(s_i,k)$
$z_i = g(s_i,k)$
$c_i = h(z_i,m_i)$

où s_0 est l'état initial et est généralement déterminé par les bits de la clé secrète **k**, **f** est la fonction d'état, **g** est la fonction produisant la suite pseudo-aléatoire z_i et **h** est la fonction de combinaison entre cette suite z_i et le clair m_i produisant le chiffré c_i. La fonction d'état **f** décrit les changements d'états internes entre deux instants. Inspirée directement des fonctions analogues des automates cellulaires, cette fonction est, dans les algorithmes modernes, confondue avec la fonction **g**.

À titre d'exemple d'algorithme possédant une fonction d'état distincte, signalons l'algorithme de chiffrement de *Mealy*. Les opérations de chiffrement et de déchiffrement sont décrites dans la figure 1 ci-contre.

Eric Filiol
Ecole Supérieure et d'Application des Transmissions
Laboratoire de cryptologie et de virologie
efiliol@esat.terre.defense.gouv.fr
http ://www-rocq.inria.fr/codes/Eric.Filiol/index.html

Figure 1 *Système de chiffrement synchrone (sigma représente s)*

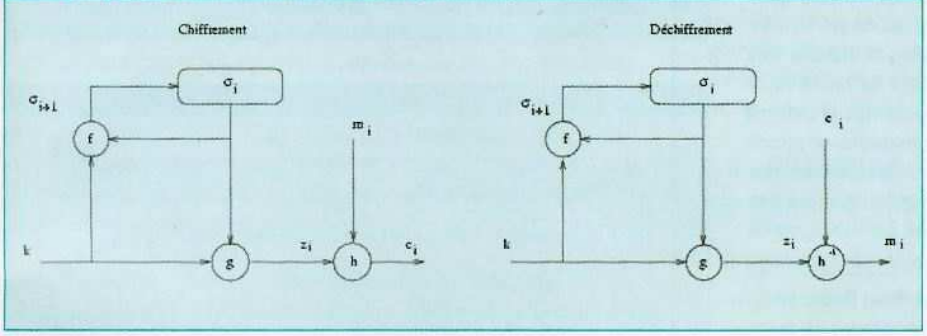


Figure 2 *Système de chiffrement binaire additif*

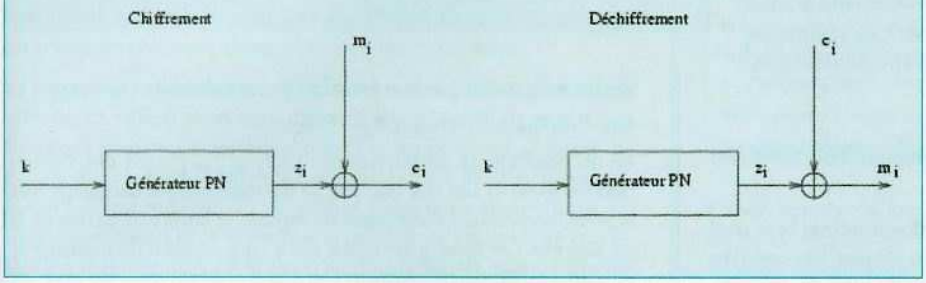
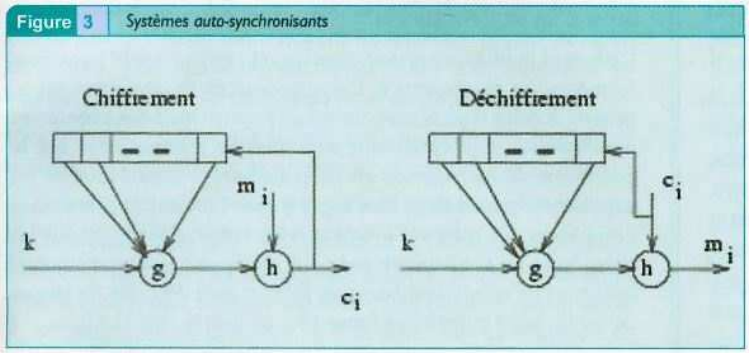


Figure 3 *Systèmes auto-synchronisants*



Dans un système synchrone, les émetteur et destinataire doivent être synchronisés (d'où le nom de synchrone), c'est-à-dire doivent utiliser à chaque temps **t** le même bit de clé à la même position de texte clair et chiffré. Toute perte de synchronisation (insertion ou délétion d'un symbole) se traduit par un déchiffrement impossible. Des techniques de resynchronisation (réinitialisation, marqueurs...) corrigent efficacement le problème. À noter que ces systèmes sont très résistants au bruit. Tout bit du texte chiffré modifié par le bruit sera mal déchiffré mais ce déchiffrement erroné ne perturbera en aucun cas celui des autres bits du texte chiffré.

La plupart des systèmes connus dans la littérature sont de ce type et plus exactement du type additif :

Définition
Un système de chiffrement *binaire additif* (figure 2) est un système par flot synchrone dans lequel la suite clé pseudo-aléatoire, la suite de texte clair et la suite de texte chiffré sont de nature binaire et où la fonction **h** est la fonction XOR.

Les systèmes auto-synchronisants (ou systèmes asynchrones)

Définition
Un système de chiffrement par flot *auto-synchronisant ou asynchrone* est un système dans lequel la suite pseudo-aléatoire est fonction de la clé et d'un nombre fixe de bits précédents du texte chiffré.

Dans la littérature industrielle, le terme de système à autoclave sur le chiffré est également rencontré. Le chiffrement s'opère par les équations suivantes :

$s_i = c_{i-t},c_{i-t+1},\ldots,c_{i-1})$,
$z_i = g(s_i,k)$ et
$c_i = h(z_i,m_i)$.

où $s_0 = (c_{-t}, c_{-t+1},\ldots, c_{-1})$ est l'état initial (non secret), **k** est la clé, **g** la fonction produisant la suite pseudo-aléatoire z_i et **h** est la fonction de combinaison entre cette suite z_i et le clair m_i produisant le chiffré c_i. La figure 3 décrit le principe général de ces systèmes.

L'avantage de ces systèmes est qu'en cas d'effacement ou d'insertion de bits dans le chiffré, l'auto-synchronisation reste possible étant donné que le déchiffrement ne dépend que d'un nombre limité de bits précédents de chiffré. Cela permet en cas de perte de synchronisation de la rétablir automatiquement. Un nombre restreint de bits sera alors perdu. La redondance de la langue permet alors de les retrouver.

La propagation des erreurs dues au bruit est de même limitée au nombre **t** de bits de chiffré utilisé pour la génération de la suite pseudo-aléatoire. Enfin ces systèmes sont plus robustes que les systèmes synchrones aux attaques utilisant les propriétés de redondance du texte clair, car chaque bit de clair influence la totalité du texte chiffré, assurant ainsi une très bonne diffusion des statistiques du clair.

Structure générale d'un système par flot

Tous les systèmes de chiffrement par flot, du moins pour la très grande majorité d'entre eux, se compose de trois modules :

→ un moteur composé le plus souvent de registres à décalage placés en parallèle, qui génère rapidement des suites de bits possédant déjà des propriétés statistiques minimales (appelées critères de Golomb [3]) ;

→ ces suites possédant encore des faiblesses, en particulier une plus ou moins grande linéarité selon la nature des registres utilisés dans le moteur, un module de haute non-linéarité aura pour fonction de prendre les suites produites par le moteur et de les transformer de manière optimale afin de gommer toute faiblesse résiduelle et exploitable par l'attaquant. Ce module est composé de fonctions booléennes combinées entre elles. En sortie, une suite binaire, appelée *suite chiffrante* est produite ;

→ un module de combinaison qui réalise l'opération proprement dite de chiffrement en combinant la suite chiffrante et le texte clair pour produire le texte chiffré. Dans la plupart des cas, il s'agit d'une simple addition modulo 2, réalisée bit à bit. De simples calculs de probabilité (théorème des probabilités totales) montrent facilement que si la suite chiffrante est aléatoire, alors, quelles que soient les statistiques du texte clair, le texte chiffré sera également aléatoire.

La figure 4 résume cette structure générale.

Notons que dans certains cas, ces modules peuvent être combinés partiellement selon le mode parallèle.

La clé secrète, dans le cas des systèmes pseudo-aléatoires (voir plus loin ce terme), sert à initialiser les primitives du moteur (les registres à décalage) et quelquefois, certaines primitives du module de haute non-linéarité. Les transformations successives s'opèrent de telle manière qu'il est impossible pour l'attaquant :

→ de déterminer la valeur du **n-ième** bit de la suite chiffrante en connaissant les **n** premiers (non prédictibilité),

→ de déterminer tout ou partie des bits de clé (autrement que par essais systématiques) à partir de la suite chiffrante réellement utilisée (en pratique, comme aucun système n'est réellement mathématiquement parfait, l'ingénieur s'assure qu'aucune attaque n'est possible sans requérir un nombre prohibitif de bits de suite chiffrante).

Figure 4 *Structure générale d'un système de chiffrement par flot*

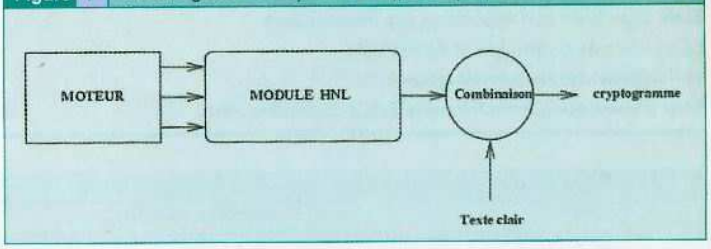
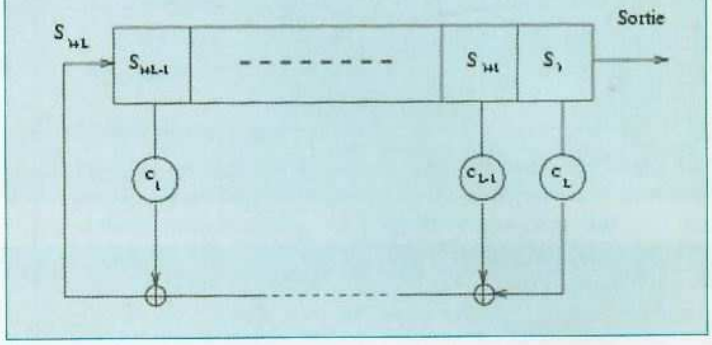


Figure 5 *Registre à décalage à rétroaction linéaire*



Les registres à décalage

Ces éléments constituent les briques de base du moteur. Le plus souvent il s'agit de registres à décalage à rétroaction linéaire, plus facile à implémenter en hardware que les autres types de registres (où la rétroaction est non linéaire). Ces registres sont aussi connus sous le nom de LFSR (*Linear Feedback Shift Register*)

Un registre à décalage à rétroaction linéaire de longueur **L** est composé d'un registre à décalage contenant une suite de **L** bits $(s_i,\ldots,s_{i+L-1})$ et d'une fonction de rétroaction linéaire (voir figure 5).

À chaque impulsion d'horloge, le bit s_i constitue la sortie du registre et les autres bits sont décalés vers la droite (ce sens est arbitraire et pourrait être inversé). La case de gauche à présent vide reçoit le bit s_{i+L}.

Ce bit est produit par une fonction linéaire des bits $(s_i,\ldots,s_{i+L-1})$:

$s_{i+L} = c_1 s_{i+L-1} + c_2 s_{i+L-2} + \ldots + c_L s_i$

où les coefficients de rétroaction $(c_i)_{1 <= i <= L}$ sont des valeurs binaires (0 ou 1). Les bits $(s_0,\ldots,s_{L-1})$ qui déterminent complètement la suite, constituent *l'état initial du registre* et sont une partie de la clé secrète. On appelle encore « *pas* » l'ensemble d'un décalage et du calcul du bit de rétroaction. Dans une configuration de base, les plus simples, un registre « avance » à raison d'un pas à la fois et produit un bit à chaque pas.

L'élément le plus important dans un tel registre est le polynôme formé par les coefficients c_i. Ce polynôme, dit *de rétroaction*, est en général linéaire d'où le nom donné au registre mais des polynômes non linéaires sont également utilisables. La théorie exige que le polynôme de rétroaction ait certaines propriétés. Nous ne les exposerons pas ici. Elles font appel à des résultats d'algèbre assez complexes. Ce qu'il est important de savoir, c'est que tous les polynômes ne conviennent pas pour un usage cryptographique et le corpus de résultats théoriques permet sans difficulté de choisir parmi les bons polynômes (pour plus de détails voir [3,4]).

Un aspect important est de s'assurer que le registre est en période maximale, c'est-à-dire que les $2^L - 1$ états internes soient tous représentés dans le cycle de transition du registre ; autrement dit, un état interne donné ne réapparaît que lorsque tous les états internes possibles ont été d'abord réalisés. Dans le cas contraire certains états seraient toujours absents, ce qui constitue une faiblesse importante.

Si les suites produites par les registres ont de relativement bonnes propriétés statistiques de base, elles présentent en revanche le grave défaut, en cryptologie, d'être linéaires. La linéarité implique un certain degré de prédictibilité exploitable par l'attaquant et surtout la capacité à modéliser tout ou partie de l'algorithme par des systèmes d'équations dont la résolution permettra à l'attaquant de retrouver, sinon la clé, du moins une partie importante de cette dernière. Il

est donc nécessaire de briser ces propriétés de linéarité. C'est le rôle du deuxième module composé essentiellement de fonctions booléennes.

Les fonctions booléennes

Il existe deux types de fonctions de ce type :

→ *les fonctions booléennes simples* qui à n variables binaires (des bits) font correspondre un seul bit. Un exemple est la fonction **f** qui aux bits (x_1, x_2, x_3) fait correspondre le bit calculé ainsi $x_1.x_2 + x_1.x_3 + x_2.x_3$ où ., + représente respectivement la multiplication et l'addition modulo 2.

→ *les fonctions booléennes vectorielles* qui à **n** bits font correspondre **m** autres bits (avec en général $m < n$). Plus compliquées à étudier, elles sont généralement vues comme un ensemble de **m** fonctions booléennes simples.

Les fonctions booléennes constituent des primitives cryptographiques essentielles dans le chiffrement à flot (et en cryptographie d'une manière générale). En fait, la plus grande partie de la sécurité d'un système de ce type repose sur ces fonctions qui le composent. C'est la raison pour laquelle la recherche sur les fonctions booléennes est très active et surtout capitale. Le problème est que les fonctions booléennes que l'on peut et l'on doit utiliser en cryptographie sont en nombre extrêmement faible.

La recherche sur les fonctions booléennes est donc extrêmement difficile. La raison tient tout d'abord à l'ensemble de toutes les fonctions possibles dans lequel chercher les bonnes. Les fonctions booléennes à **n** variables sont au nombre de 2^{2^n}. Par exemple, si $n = 6$, il faut explorer 2^{64} fonctions. L'approche énumérative est donc impossible. De plus, dès que le nombre de variables en entrée devient trop grand, il est impossible de stocker et donc d'étudier ces fonctions directement en travaillant sur l'ensemble de leurs valeurs (un tel ensemble aurait une taille de 2^n pour une fonction à **n** variables). L'étude des fonctions booléennes ne peut donc se faire que grâce à des mathématiques très évoluées et malheureusement trop complexes pour être évoquées ici.

L'enjeu est de trouver des fonctions qui possèdent le plus grand nombre, simultanément, de propriétés cryptographiques fortes :

→ l'équilibre, c'est-à-dire que la fonction doit produire autant de 1 que de 0 sur l'ensemble de ses 2^n entrées,

→ une haute non linéarité ; *grosso modo* la fonction booléenne doit être le plus éloignée (au sens d'une distance que nous ne définirons pas ici) de toutes les fonctions linéaires existantes et à **n** variables. Pour plus de détails, consulter [4].

→ l'immunité aux corrélations d'ordre **t** ; il faut interdire l'existence de corrélations entre d'une part, toute somme d'au plus **t** variables d'entrées, et d'autre part la sortie de la fonction (une corrélation est une probabilité différente de 0.5 dans notre cas). Quand de telles corrélations existent, il est alors possible d'obtenir de l'information sur l'entrée de la fonction en considérant uniquement sa sortie. Comme il existe toujours des corrélations (théorème de Parseval), il faut choisir des fonctions n'autorisant que des corrélations d'ordres élevés (autrement dit l'information obtenue à partir de la sortie concerne un trop grand nombre simultanément de variables d'entrées pour permettre d'exploiter l'information).

Le principal problème tient au fait que depuis les années 80, la théorie a montré que ces propriétés s'opposent et que seuls des compromis sont possibles. C'est là que l'art de l'ingénieur prend le pas sur celui du mathématicien. Ces compromis sont en fait surtout dictés par l'implémentation, c'est-à-dire l'usage pratique de ces fonctions dans un schéma réel. Les fonctions booléennes ne peuvent donc jamais être utilisées seules, comme c'est le cas dans les systèmes proposés dans la littérature et qui vont être présentés dans la section suivante. L'ingénieur va au contraire les associer de manière efficace et subtile, selon différentes techniques, de manière à contourner les limitations opposées par les fonctions booléennes et les schémas de base existants.

Les différents schémas de base de chiffrement par flot

Le système de Vernam

L'origine de ce chiffrement provient du système de Vernam [1].

Définition

Le chiffrement de Vernam est un système de chiffrement par flot sur l'alphabet $A = \{0, 1\}$. Un message binaire $m_1m_2m_3...m_t$ est combiné avec une suite-clé binaire $k_1k_2k_3...k_t$ de même longueur pour produire une suite chiffrée $c_1c_2c_3...c_t$ où $c_i = m_i + k_i$ avec $1 <= i <= t$ (le signe + ici symbolise l'addition modulo 2).

Quand la suite clé est générée indépendamment et aléatoirement, on parle alors de chiffrement par bande aléatoire une fois (*One time pad* ou encore masque jetable). Ce type de chiffrement a été prouvé parfaitement sûr contre toute attaque, même à chiffré seul (troisième théorème de Shannon traitant du secret parfait). On parle alors de sécurité inconditionnelle (*grosso modo*, cela peut se résumer en disant que même si l'attaquant disposait d'une puissance de calcul infinie et de l'éternité pour la mettre en œuvre, il ne parviendrait pas, malgré cela, à déterminer quel texte clair et quel clé ont été utilisés).

Plus précisément, si **M**, **C** et **K** sont des variables aléatoires désignant respectivement le texte clair, le texte chiffré et la suite clé (secrète), alors $H(M) = H(M|C)$ où $H(.)$ désigne la fonction entropie (cette fonction sert à mesurer le degré d'incertitude concernant une quantité inconnue ; plus précisément, pour une variable aléatoire discrète **X** prenant les valeurs $X_1, X_2, \ldots, X_n$ avec les probabilités non nulles $p_1, p_2, \ldots, p_n$, alors $H(X) = -(p_1\log_2(p_1) + p_2\log_2(p_2)... + p_n\log_2(p_n)$.

De façon équivalente, cela revient à dire que la *transinformation* (ou information mutuelle) est nulle. Autrement dit la connaissance du chiffré ne fournit aucune information sur le clair.

En pratique, des suites sont générées par des moyens physiques (résistance thermique échantillonnée par exemple) puis dupliquées avant leur mise en place chez l'émetteur et le destinataire.

Shannon [2] a prouvé qu'une condition de sécurité parfaite était que $H(K) <= H(M)$, c'est-à-dire que l'incertitude sur la clé secrète doit être au moins aussi grande que celle sur le clair. Si la clé est de longueur **k** et si les bits sont choisis aléatoirement et indépendamment les uns des autres, alors $H(K) = k$ et la condition

de Shannon devient **k <= H(M)**. La bande aléatoire une fois est inconditionnellement sûre, quelle que soit la distribution statistique des bits de clair.

Le problème majeur de ce schéma est que la clé doit être de longueur aussi grande que celle du texte clair à chiffrer. Cela complique extrêmement la génération et la gestion des clés. Il est important de rappeler que l'utilisation répétée d'une même bande (pour plusieurs messages différents, appelés alors messages parallèles) est catastrophique et permet en quelques secondes de retrouver la suite clé et les messages. D'où le terme « *une fois* » traduisant l'obligation de ne pas réutiliser la même bande.

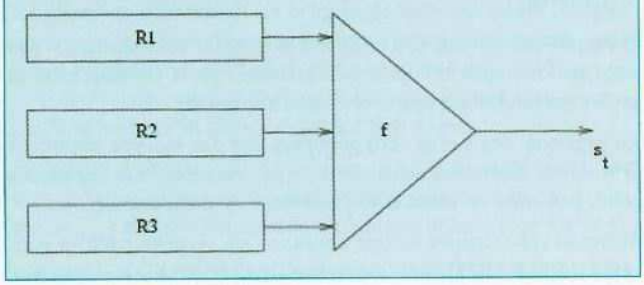
C'est la raison pour laquelle, les systèmes de type Vernam sont réservés à des communications d'un très haut niveau de sensibilité (stratégique). Notons que la cryptographie quantique travaille essentiellement sur la conception de systèmes et de protocoles permettant la mise en place efficace et sécurisée de part et d'autre d'une communication de bandes aléatoires une fois (protocole Brassard-Bennett par exemple).

Ce problème a motivé la conception de chiffrement par flot où la suite clé est *pseudo-aléatoire*, c'est-à-dire générée de façon déterministe à partir d'une clé beaucoup plus petite (il s'agit d'une suite reproductible donc dans l'absolu non réellement aléatoire ; mais cette suite est conçue de manière à ressembler le mieux possible à un aléa véritable). Comme, dans ce cas, **H(K)** est très petit par rapport à **H(M)**, le système n'est plus à secret parfait et ne respecte plus la théorie de Shannon sur le secret parfait. Une manière simple de s'en persuader vient du fait que si la clé a une entropie de **k** bits (autrement dit la taille de la clé est de **k**, en pratique), il « suffit » d'essayer les 2^k clés possibles pour retrouver la bonne clé utilisée. Toutefois, on s'assure lors de la conception que la sécurité soit suffisante pour l'usage voulu (i.e. temps de calcul nécessaire à l'attaquant prohibitif). En particulier, la taille **k**, en bits, de la clé doit être suffisamment importante pour interdire calculatoirement toute recherche exhaustive sur la clé. Nous allons maintenant étudier les principales primitives de base (ou leur macro assemblages) qui constituent ces systèmes. Comme la quasi-totalité des systèmes réels utilisent la fonction `xor` comme fonction de combinaison avec le texte clair, nous nous intéresserons uniquement au moteur et au module de non-linéarité.

Les systèmes à combinaison de registres

Ce système constitue le système de chiffrement à flot le plus simple qui soit.

Figure 6

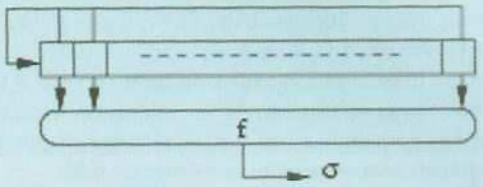


Dans ce type de système, la suite $(s_t)_{t>=0}$ issue de la fonction **f** est générée à partir des suites d'entrée produites par plusieurs registres à décalage. L'avantage de ce schéma est d'augmenter considérablement l'espace des clés. En effet, la clé secrète est alors constituée non plus de l'initialisation d'un seul registre mais de plusieurs.

Bien que la littérature fasse de ce type de schéma un système de chiffrement à lui tout seul, en réalité il n'est que très rarement utilisé ainsi dans les systèmes réels réclamant un haut niveau de sécurité. Cela est en grande partie dû à l'impossibilité de trouver des fonctions booléennes ayant les propriétés requises. En particulier, il sera toujours possible d'exhiber une corrélation entre un sous-ensemble d'entrées et la sortie de la fonction, livrant ainsi une information sur la clé (le contenu des registres). Nous parlerons des attaques un peu plus loin.

Une variante de ce schéma est le registre filtré, décrit par le schéma suivant :

Figure 7 *Schéma de registre filtré*



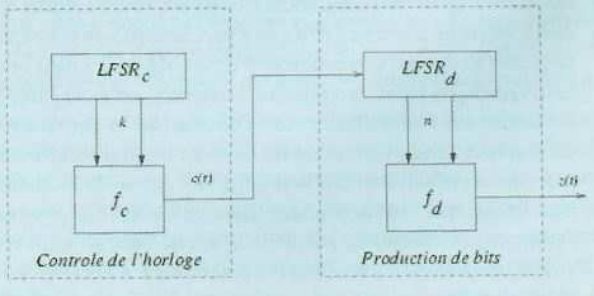
Un unique registre est utilisé, de longueur **L**, produisant une suite en période maximale. Certaines cellules de ce registre sont prises en entrée par une fonction **f** non linéaire à **n** variables. C'est la sortie de la fonction **f** qui va constituer la suite pseudo-aléatoire qui sera soit combinée directement avec le texte clair soit, le plus souvent, utilisée en entrée d'une autre brique de base

T. Siegenthaler a montré en 1986 que ce schéma peut toujours se ramener à un schéma à combinaison de registres utilisant une fonction de combinaison **g** à **m** variables (avec **1 < m <= n**). Les registres ont alors tous le même polynôme de rétroaction. Seuls les états initiaux diffèrent.

Les schémas à horloge contrôlée

Comme pour les schémas à décimation, les schémas à horloge contrôlée visent à supprimer tout lien aisé à décrire entre les bits de produits par le schéma et l'état interne des registres à un quelconque temps. La figure suivante résume la philosophie de ce type de schéma :

Figure 8 *Système à horloge contrôlée (LFSR = registre)*



Dans une configuration de base (qui peut être rendue plus complexe en considérant plus d'éléments), deux registres à décalage (en général à rétroaction linéaire) sont utilisés :

→ un premier registre dit « *de contrôle d'horloge* » est régulièrement mis à jour (le registre avance d'un pas à chaque coup d'horloge). La suite de bits produite par ce registre contrôle la mise à jour du sous-système de génération de bits. Supposons, à titre d'exemple que les bits produits soient considérés par paires : (b_i, b_{i+1}).

→ un second registre dit « *de production de bits* ». L'avancée de ce registre (autrement dit le nombre de pas) est directement déterminée par la valeur des bits produit par le registre précédent. Les deux bits (b_i, b_{i+1}) décrivent un nombre c_t compris entre 0 et 3. Ce nombre représente donc le nombre de pas (décalage + calcul du bit de rétro-action) avec lequel avance le second registre avant de produire un bit de la suite finale.

Avec ce type de schéma, non seulement les bits de clé (i.e. l'état interne initial des registres) sont des variables aléatoires mais également la valeur d'avancée du second registre. Le but est d'accroître l'incertitude de l'attaquant et donc de rendre l'attaque plus difficile.

Les schémas à décimation

Comme pour les schémas à horloge contrôlée, le but de ces schémas est de compliquer la tâche de l'attaquant en supprimant une autre donnée habituellement facilement disponible : l'indice de temps. Dans les schémas à horloge contrôlée, la donnée supprimée est la valeur du pas d'avance du second registre.

La structure générale basique de ces schémas est la même que pour les schémas précédents. La différence réside dans le fait que selon la valeur de certains bit produits par le premier registre, le bit produit par le second (qui est ou non à avance régulière, tous les cas sont possibles) est conservé ou non dans la suite finale produite. D'où le terme de décimation. Dans ce dispositif, l'attaquant ne sait plus quand un bit donné de la suite a été réellement produit. L'indice de temps liant ce bit et l'état interne des registres a été supprimé.

Les schémas à mémoire

Dans ce type de systèmes, le concepteur cherche à accroître l'effet de chaque bit de clé sur les bits de sortie en considérant un mécanisme de mémoire : l'état interne de certaines primitives conservent pendant quelques instants, la valeur de certaines variables afin de prolonger leur action. Deux grandes familles existent, à côté de très nombreuses variantes :

→ les schémas « *avec retenue* ». Dans tous les autres schémas, l'opération de base utilisée est l'addition modulo 2. Cette dernière est une opération dite sans retenue (les opérandes et le résultat ne peuvent que valoir 0 ou 1, puisque tout est défini sur une structure algébrique à deux éléments). Si au lieu de travailler en binaire, on considère cette fois-ci des entiers en travaillant avec l'addition conventionnelle sur les entiers, apparaît alors le phénomène de retenue classique. La valeur de cette dernière est mémorisée dans certains états internes du schéma et intervient sur la suite des opération comme n'importe quelle autre variable. Ce faisant, l'influence de plusieurs bits (vus cette fois comme des entiers) est prolongée par ce mécanisme de retenue. Un exemple célèbre est celui de l'algorithme **E0**, présent dans la norme *Bluetooth* pour le chiffrement des données (l'addition considère des entiers compris entre 0 et 3). Les structures avec générateurs congruentiels peuvent être rattachées à cet ensemble (voir [0] pour plus de détails).

→ les schémas avec « *blocs mémoire* » (exemple le générateur de *Marsaglia-McLaren*). Dans cette configuration, une structure à mémoire reçoit de manière continue des valeurs transformées ou non des bits de clés (un grand nombre de variantes est possible pour le remplissage de cette structure). Les bits de cette structure sont alors utilisés de manière aléatoire. Autrement dit, l'indice de temps **t** décrivant l'instant où un bit donné de la structure est réellement utilisé, devient une variable aléatoire. Dans certains schémas, le bit issu de la structure peut y être réinjecté après utilisation pour accroître l'effet de mémoire.

Les schémas chaotiques

Les systèmes à base de chaos utilisent des fonctions mathématiques différentes des registres pour générer des suites de bits. Ces fonctions ont des propriétés statistiques intéressantes tout en possédant des caractéristiques, notamment de non prédictibilité (le chaos) très intéressantes. Le principe de base est celui qui veut que le système se comporte de manière totalement différente si les conditions initiales viennent à varier, même de manière infime.

L'exemple le plus célèbre est celui de la fonction de Hénon, proposée comme primitive chaotique pour le chiffrement par Réjanne Forré en 1991.

La fonction est la suivante : on considère un couple de nombre réels (x_0, y_0) (choisis dans un domaine de définition adéquat) et la fonction de récursion suivante :

$x_{n+1} = 1 + y_n - 1{,}4(x_n)^2$,
$y_{n+1} = 0{,}3x_n$

À chaque instant **n**, un bit z_n est produit de la manière suivante :
$z_n = 0$ si $x_n <= 0{,}39912$
$z_n = 1$ sinon.

Cette fonction, malheureusement, présente certaines faiblesses car le motif **1100** n'apparaît jamais dans la suite produite. Il est cependant possible en considérant deux fonctions de Hénon utilisées en série d'obtenir une primitive chaotique ayant toutes les qualités requises.

De très nombreux autres systèmes chaotiques ont été proposés. Il serait trop long de tous les citer ici. À ce jour, beaucoup ont été prouvés comme n'offrant pas toutes les garanties de sécurité des modèles classiques. Cela ne signifie aucunement que cette voie de recherche soit à dédaigner, bien au contraire. Des systèmes récemment proposés comme ceux de l'ENSEA, en France [5], fondée sur des fonctions chaotiques plus complexes (générateur hyperchaotique de Roëssler) apparaissent comme très prometteurs en terme de sécurité (ils sont à mi-chemin entre systèmes par blocs et systèmes par flot).

Le principal problème avec les primitives chaotiques vient du fait qu'elles travaillent non plus sur des ensembles discrets (autrement dit contenant un nombre fini d'éléments, typiquement 0 et 1) mais sur des nombres réels. Or, dans un contexte informatique, si 0 et 1 sont deux objets faciles à représenter, cela n'est plus vrai pour des nombres réels. La précision est un facteur délicat à gérer et il

est alors indispensable de recourir à des librairies spécialisées de type **GMP** (*GNU Multiprecision Package*) pour assurer compatibilité et portabilité dans le cadre d'application logicielle.

La cryptanalyse des systèmes par flot

Il existe une très importante littérature traitant de la cryptanalyse des systèmes pas flot, du moins des systèmes connus. Mais force est de reconnaître (notamment lorsque l'on reprogramme les attaques proposées) que rares sont les techniques réellement efficaces et opérationnelles.

Deux grandes classes d'attaques sont connues (nous n'évoquerons pas les techniques exotiques exploitant des faiblesses structurelles de certains systèmes) :

→ celles exploitant la notion de corrélation, autrement dit l'existence d'une information de nature probabiliste entre la sortie et l'entrée de certaines primitives (attaque par corrélation, par corrélation rapide, par décodage...). Ces techniques, pour être efficaces (lorsqu'elles le sont), nécessitent de disposer d'une quantité trop importante de texte chiffré – ou de texte clair pour certaines attaques – (plusieurs milliards de bits) pour être opérationnelles. La gestion de la clé surtout dans certains contextes d'utilisation (téléphonie mobile par exemple) requiert de changer la clé très souvent, rendant par conséquent ces attaques pratiquement impossibles.

→ les techniques algébriques de résolution d'équations. Ces techniques sont récentes et leur faisabilité ne fait pas l'unanimité. Leur principal défaut est une complexité trop grande pour être réellement mise en œuvre de manière opérationnelle. À ce jour aucune cryptanalyse réellement efficace n'a été publiée.

Dans tous les cas, ces techniques sont toujours présentées pour des systèmes très basiques – se réduisant à l'un des schémas de base présentés plus haut – très éloignés des systèmes réels. Un moyen simple de lutter de manière efficace contre ces attaques est l'usage de la décimation ou du contrôle d'horloge afin d'interdire ou de très fortement limiter la possibilité d'exhiber des équations de degré 1 liant des bits de sortie avec des bits de clés. En effet, ces deux grandes classes d'attaques ne fonctionnent qu'en disposant d'un très grand nombre de telles équations.

L'absence de challenges sérieux de cryptanalyse (taille de séquence nécessaire et temps de calcul réalistes) permettant de vérifier la validité des attaques présentées ne contribue certainement pas à clarifier le débat. Encore trop d'incertitudes, de fantasmes, voire d'idées fausses, subsistent concernant la sécurité ou l'insécurité prétendue des systèmes par flot.

Pour résumer, à côté de l'attaque par force brute (essayer toutes les 2^n clefs possibles pour un système fonctionnant avec une clef de n bits), ne sont connues que des attaques théoriques inefficaces dans un contexte réel d'emploi. Ces attaques requièrent soit beaucoup plus de bits interceptés (de la suite chiffrante) qu'aucun message n'en contiendra jamais en pratique, soit sont d'une complexité telle qu'elles restent irréalisables alors même qu'elles font mieux qu'une attaque par force brute. Des systèmes très utilisés, comme **RC4** ou **E0** (norme *Bluetooth*) n'ont jamais été réellement remis en cause et sont toujours en usage.

Enfin, et cela est capital, n'oublions jamais qu'un système cryptologique (quel qu'il soit) est toujours construit par rapport aux cryptanalyses connues et/ou publiées. Cela signifie, entre autres aspects, que le temps et avec lui le recul théorique de l'ingénieur est capital. De ce point de vue, le chiffrement par flot, dont les premières bases ont été jetées il y a près de quatre-vingts ans, possède sur les autres technologies de chiffrement un indéniable avantage.

Conclusion

Les cryptanalyses publiées sur ces systèmes ne considèrent que des briques de bases qu'aucun ingénieur cryptologue sérieux n'utiliserait telles quelles. Ces résultats ont contribué à jeter le discrédit sur une technologie qui continue de faire ses preuves pour des besoins nécessitant un très haut niveau de sécurité opérationnelle. Les schémas de base présentés dans cet article doivent être combinés de manière astucieuse pour contourner les limitations avancées par la théorie mathématique. C'est cet aspect qui certainement rend les systèmes par flot à la fois si mal perçus et si intéressants. La part de l'esprit humain est encore capitale.

L'autre aspect qu'oublient souvent les détracteurs des systèmes par flot est que toute solution de cryptographie ne peut ni ne doit se résumer à adopter un algorithme robuste. La partie dite chiffre – c'est-à-dire les règles d'emploi du système, de la gestion des clés... – est fondamentale. Aucun officier du chiffre (la personne dirigeant une cellule du chiffre) n'autorisera le chiffrement de longs messages sans changer la clé. De même, la plupart des protocoles de transmissions travaillent sur la base de trames courtes (de l'ordre de la taille de la clé). L'aspect gestion des clés devient alors fondamental. Se doter d'un système, aussi robuste soit-il, sans considérer l'ensemble de ces règles, reviendrait à installer *une porte blindée sur un mur en carton*. L'attaquant passera par le mur plutôt que par la porte. Là est l'intérêt des systèmes de chiffrement par flot car la philosophie de conception oblige dès le départ à prendre toutes ces règles en compte.

Références

[0] A.J. Menezes, P.C. Van Oorschot, S.A. Vanstone - *Handbook of Applied Cryptography*, CRC Press, 1997.

[1] G.S. Vernam - « Cipher printing telegraph systems for secret wire and radio telegraphic communications ». *Journal of The American Institute for Electrical Engineers*, Vol. 55, pp 109-115, 1926.

[2] C.E. Shannon - « A mathematical theory of communication ». *Bell System Journal*, Vol. 27 pp. 379-423 (Part I) et pp. 623-656 (Part II), 1948.

[3] S.W. Golomb - *Shift Register Sequences*. Aegean Park Press, 1982.

[4] E. Filiol - *Techniques de reconstruction en cryptologie et en théorie des codes, thèse de doctorat*, Ecole Polytechnique, 2001. Disponible sur **http ://www.inria.fr/rrrt/tu-0671.html**.

[5] I. Belmouhoub, M. Djemai et J.-P. Barbot - *Cryptography by discrete-time hyperchaotic systems*, 42nd Conference on Decision and Control, pp. 1902-1906, 2003.

détection) pour les réseaux informatiques, un dispositif qui couperait les connexions non souhaitées de type « données ». Cet équipement n'existe malheureusement pas en tant qu'élément optionnel d'un PABX.

Le meilleur moyen d'éviter la création d'un court-circuit par un utilisateur est certainement de lui offrir les moyens répondant à ses besoins, mais de manière contrôlée. Il est évident qu'interdire l'accès à l'Internet à ses usagers augmentera les chances que ces derniers tentent de contourner cet interdit.

Une alternative technique est de n'utiliser en interne que des interfaces numériques, aussi bien pour les téléphones que les fax, afin de rendre inutilisables les modems analogiques. Les protocoles numériques des PABX étant bien souvent différents du RNIS, les usagers seront dans l'impossibilité d'utiliser un modem RNIS pour établir des connexions. L'inconvénient de cette solution est bien évidemment son coût plus élevé que l'utilisation de postes analogiques. Au passage, il est important de signaler que le protocole analogique d'un PABX est lui aussi enrichi par rapport au système analogique utilisé par les particuliers. Pour cette raison, s'il est possible d'utiliser un poste analogique classique (et donc un modem) sur un PABX, ce dernier ne pourra pas profiter de toutes les fonctionnalités du PABX, faute de signalisation adéquate.

L'authentification

L'authentification sur un PABX est limitée par le fait que l'usager ne dispose la plupart du temps que d'un clavier téléphonique. De plus, le poste téléphonique n'étant généralement pas multi-utilisateurs, l'utilisateur et son téléphone ne font qu'un du point de vue de la configuration du PABX. Les authentifiants de l'usager sont donc des codes PIN entrés par le combiné. Or ce mécanisme d'authentification est utilisé pour protéger l'ensemble des fonctions accessibles via un poste téléphonique. Tout le monde pense naturellement à l'accès à la boîte vocale, mais il est possible, selon les autocommutateurs et leur configuration, de réaliser beaucoup d'autres opérations à partir des postes téléphoniques. Les boîtes vocales peuvent par exemple être lues et configurées à partir d'autres postes que celui auquel est liée la boîte vocale ; il suffit pour cela de demander la boîte vocale voulue et de s'authentifier par un code PIN.

D'autres fonctions sont également protégées par un code PIN, comme le déverrouillage du poste téléphonique. Ceci est utile afin d'éviter que votre poste serve d'outil à un manipulateur en votre absence. Enfin, certains autocommutateurs compacts sont parfois entièrement configurables via un téléphone, l'administrateur naviguant alors dans des menus vocaux après s'être authentifié par un code PIN.

Il faut bien réaliser qu'un code PIN est bien plus faible en termes de sécurité qu'un mot de passe, car l'alphabet est très réduit et que la longueur par défaut, bien qu'elle puisse être augmentée par l'administrateur, est faible (4 chiffres). À cela s'ajoute l'utilisateur final qui, s'il a changé son code d'accès par défaut (0000, 1111, etc.), ce qui reste très rare, l'aura remplacé par un numéro mnémotechnique (année de naissance par exemple). Enfin, il existe rarement une notion de blocage de compte après un nombre d'essais infructueux. Même si les codes PIN sont stockés sous forme « chiffrée », il est relativement facile de voir que la majorité des postes possèdent le même condensat, c'est-à-dire le mot de passe par défaut.

Les verrous logiciels et/ou matériels

Le PABX est géré par un applicatif fortement modulaire. De fait, les applicatifs sont tous présents sur le système, sauf s'ils nécessitent des éléments matériels particuliers, mais la plupart des éléments optionnels ne sont pas activés faute de posséder le verrou logiciel et/ou matériel (*dongle*) correspondant dont le fonctionnement varie selon le constructeur.

Ce verrou est en fait l'équivalent des clés d'activation de certains logiciels et dans le cas d'un dongle matériel, sa présence est régulièrement contrôlée. Il est possible de regarder l'ensemble des verrous utilisés par l'intermédiaire des interfaces d'administration en analysant les paramètres d'installation (Alcatel 44xx) ou du système (EADS M65xx), rubrique « verrou logiciel ». Lors d'une exploitation locale du PABX (sur le système lui-même), les commandes `spadmin` (Alcatel) ou `xdongle` (certains EADS M65xx) permettent d'obtenir les mêmes informations.

Parmi l'ensemble des verrous logiciels installés, il est possible de repérer rapidement si des fonctionnalités dangereuses sont présentes ou non. En revanche, comme tout mécanisme à base de clé d'activation, il ne peut être exclu qu'une personne réussisse à activer des fonctions à l'insu de l'administrateur en fournissant les verrous adéquats.

Une fonctionnalité particulièrement dangereuse qui nécessite un verrou est la DISA. Cette fonction particulière permet à une personne extérieure à l'entité de se connecter au PABX, puis d'hériter des droits d'un poste interne sous réserve d'authentification. L'origine historique de cette fonctionnalité est d'offrir une prise en charge des communications des nomades d'une entreprise. Un nomade contacte un numéro, parfois vert, correspondant à la DISA, s'authentifie en fournissant le numéro de poste interne ainsi qu'un code PIN : il peut alors passer des communications qui sont à la charge de la société. Inconvénient de cette fonctionnalité, elle est optionnelle et inutile, mais elle se retrouve cependant installée sur un grand nombre d'équipements.

La prolifération des flottes de téléphones portables aurait dû sonner le glas de cette fonctionnalité que l'on rencontre encore. Bien qu'il existe des processus d'authentification avancés (type « SecurID »), la sécurité repose souvent sur la définition du code PIN par l'utilisateur qui placera par exemple une date de naissance... De plus, le numéro SDA affecté à la DISA doit bénéficier des mêmes soins que celui de la télémaintenance.

Cette fonctionnalité constitue une cible privilégiée des phreakers, car elle permet d'effectuer aux frais de la victime des appels lointains si un accès à un compte intéressant est découvert. L'activation de cette fonction doit être évitée et dans le cas contraire, ne doivent pas être attribués aux utilisateurs de la DISA des droits permettant d'effectuer des appels coûteux.

La gestion des droits d'établissement de communications

Derrière ce terme, qui en fait varie selon le constructeur du PABX, se cache la restriction de droits la plus connue de tous les utilisateurs du téléphone : ai-je le droit au local, national ou, plus chanceux, à l'international ? Étant donné qu'un PABX peut gérer plusieurs milliers

- JSR 82 : Java API's for Bluetooth
- JSR 135 : Mobile Media API (MMAPI) 1.1
- JSR 172 : J2ME Web Services API
- JSR 184 : Mobile 3D Graphics
- JSR 205 :Wireless Messaging API (WMA) 2.0

4 Des API spécifiques constructeur : dans ce domaine tout est permis ...

De nombreux environnements de développement existent pour le développement de code Java sur téléphones portables :

→ Des solutions gratuites telles que le J2ME Wireless Toolkit [8], téléchargeable sur le site de Sun mais également proposé en version « customisée » par de nombreux constructeurs tels que Nokia, Ericsson, etc. (figure1).

→ Des solutions payantes chez tous les grands éditeurs d'environnements de développement, tels que Sun One Studio Mobile Edition, Borland JBuilder Mobile Edition, CodeWarrior Wireless Development Kit, etc.

La documentation est quant à elle librement consultable sur le site de Sun [9]. Tout ceci contribue à rendre le développement Java sur téléphones portables simple et accessible. Le traditionnel « Hello, World » en Java est disponible à l'adresse suivante : **http://developers.sun.com/techtopics/mobility/midp/articles/getstart/**

L'application est très proche d'une applet Java traditionnelle, à l'exception des imports (`import javax.*`, voire `import com.nokia.*` ou autres pour les appels plus spécifiques à la plate-forme sous-jacente), et des points d'entrée (`startApp()`, `pauseApp()`, `destroyApp()`). Toutes les applications dérivent de la classe `MIDlet`.

Symbian OS

Le développement Symbian OS nécessite un environnement de développement standard (de type GCC, Visual Studio, MetroWerks CodeWarrior, Borland C++ Builder X ou autre) et le SDK Symbian, disponible gratuitement sur le site Symbian [10].

Les deux principaux constructeurs utilisant Symbian OS fournissent également des versions « adaptées » du SDK, qui gèrent les spécificités de leurs téléphones. En effet, le système Symbian OS est fortement « modulable », chaque sous-système pouvant être réécrit (cf. [11]).

Du coup, une bonne partie de l'interface utilisateur a été personnalisée par les fabricants de téléphones, selon des spécifications partiellement incompatibles ... On le constate lors de l'édition de liens, qui fait intervenir un grand nombre de librairies « constructeur ».

Parmi les SDK « adaptés » disponibles actuellement, on trouve :

→ UIQ (SonyEricsson P800/P900, Motorola A920)

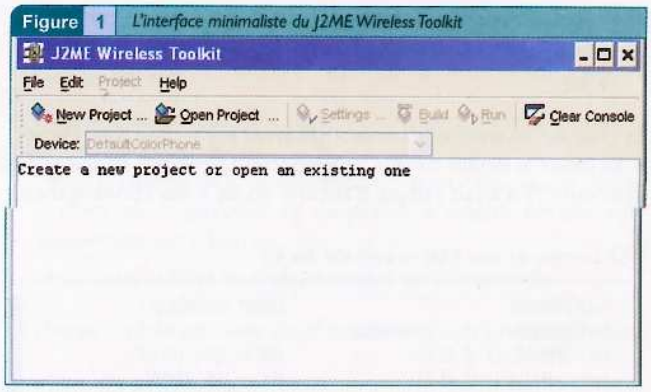
→ Nokia Série 60

→ Nokia Série 80

→ Nokia Série 90

Personnellement, je trouve les SDK Nokia plus simples d'installation et d'utilisation, les SDK Sony Ericsson ne supportant que des

Figure 1 *L'interface minimaliste du J2ME Wireless Toolkit*



environnements de développement payants (Metrowerks CodeWarrior et Borland C++ Builder).

Tous les outils traditionnels (émulateur, débogueur à distance, éditeur de ressources) sont présents dans le SDK. Le développement Symbian OS reste toutefois complexe, aussi bien dans la configuration de la chaîne de compilation que par la taille d'un simple « Hello World ».

Windows CE

Le développement Windows CE s'effectue préférentiellement avec Visual Studio Embedded. Les utilisateurs de Visual Studio retrouveront un environnement et des API analogues au Windows « classique », ce qui facilite grandement le portage d'applications.

Tous les outils traditionnels (émulateur, débogueur à distance, éditeur de ressources) sont présents dans le pack de développement.

Revue des risques potentiels

ARM

En ce qui concerne la possibilité de créer du *shellcode* en assembleur ARM pour téléphone portable, il faut savoir que les instructions LDM/STM (correspondant aux POP/PUSH) peuvent fonctionner dans 4 modes différents.

- **Full Ascending (FA)** : les adresses de pile sont croissantes et le registre SP pointe vers le dernier élément de la pile.
- **Full Descending (FD)** : les adresses de pile sont décroissantes et le registre SP pointe vers le dernier élément de la pile.
- **Empty Ascending (EA)** : les adresses de pile sont croissantes et le registre SP pointe vers le premier emplacement libre de la pile.
- **Empty Descending (ED)** : les adresses de pile sont décroissantes et le registre SP pointe vers le premier emplacement libre de la pile.

Le désassemblage d'un code créé pour Nokia N-Gage avec le SDK 1.2 en mode ARM laisse à penser que la pile est en mode FD, ce qui facilite l'exploitation de possibles « *buffer overflows* ». Il reste toutefois de nombreux problèmes à régler, tels que :

→ L'adresse de retour stockée dans R14 ne peut pas être écrasée, seule l'adresse de retour de l'appelant peut être éventuellement écrasée (comme sur SPARC).