Alexandre Buge

Epitech 5 Promo 2004

Skyrecon Systems

WiViser : une solution de sécurité intégrée

Confidentiel

Sujet : Répondre aux problèmes de sécurité pausés par l'informatique moderne en créant une solution de sécurité embarquée, et en la promulguant au sein des différents acteurs du marché de l'informatique.

Stage du 01/03/2004 au 31/08/2004

1 Sommaire

2 Résumé

Le but de ce rapport de stage est d'expliquer comment la société Skyrecon a décidé de se lancer dans la sécurité informatique, et de présenter mon travail au sein de cette entreprise.

Dans un premier temps, je décrirai l'architecture de cette jeune entreprise dont le but a été de répondre aux nouvelles problématiques de sécurité dont je soulignerai les différents points en revenant sur l'historique de l'évolution des technologies et du marché dans ce domaine.

Dans un deuxième temps, je définirai dans le détail ma tâche au sein de Skyrecon, qui a consisté à analyser et à concevoir une partie de sa solution de sécurité.

Tout d'abord son architecture modulaire gravitant autour d'un noyau et d'un ensemble d'API multi-plateformes et d'agents autonomes.

Le serveur de centralisation de politique de sécurité.

La sonde WiFi pour détecter les attaques sur ce type de support.

Et enfin un firewall réseau pour Windows ou j'expliquerai les problématiques du développement noyau ainsi que l'architecture réseau des systèmes Windows de Microsoft.

Dans un troisième temps j'illustrerai mon rôle dans la promulgation de la solution au sein du marché de l'informatique.

Pour finir, je conclurai sur l'apport du stage en terme d'expérience professionnelle à ma formation continue.

3 Présentation de l'entreprise

3.1 Le secteur d'activité

Skyrecon est un éditeur de logiciel.

Le secteur d'activité de Skyrecon Systems est la sécurité informatique au sens large du terme : c'est-à-dire, aussi bien la défense des systèmes d'information des entreprises que la pérennité des informations contenues dans leurs systèmes.

En effet, Skyrecon développe pour cela deux logiciels complémentaires. Tout d'abord WiViser, qui est un logiciel de défense intégré des postes clients. Et DataViser, qui est une solution de sauvegarde répartie automatique.

3.2 L'entreprise

Skyrecon est une jeune société anonyme au capital de 37 000 euros, créée en Novembre 2003. Elle a été fondée par 14 co-actionnaires travaillant dans les différents secteurs d'activité de l'entreprise. Elle est présidée par Monsieur Ravy Truchot, le fondateur du projet.

L'effectif de Skyrecon est actuellement composé de trois commerciaux, d'une personne pour le marketing, et d'une équipe de développement de dix personnes.

L'équipe de développement est composée de trois divisions managées par un directeur technique :

La division recherche et développement, qui est chargée de développer les couches basses des technologies de Skyrecon et d'implémenter les fondations des systèmes de sécurité en fonction des technologies émergeantes.

La division de développement d'interface graphique : qui s'occupe de la mise en forme et du développement des aspects graphiques des interfaces des logiciels de Skyrecon.

Et enfin la division intelligence artificielle, qui est chargée du développement de toutes les parties de logiciel relatives à l'analyse et la corrélation d'information, l'analyse comportementale, mais également sur les algorithmes permettant l'optimisation des traitements lourds.

Son siège social est situé au dix, rue du Colisée, dans le huitième arrondissement de Paris. Les locaux du centre de recherche et de développement se trouvent au huit, rue d'Argenteuil, dans le premier arrondissement de Paris.

Skyrecon est en étroite collaboration avec un cabinet d'avocats pour les questions juridiques, mais aussi pour les problèmes de licences des logiciels et les brevets relatifs aux technologies développées.

Dans le cadre du domaine administratif, Skyrecon travaille également avec un cabinet de secrétariat et une agence de comptabilité.

3.3 Le service

Skyrecon est un éditeur de logiciel de sécurité pour entreprise.

Les services proposés par cette société comportent :
- La recherche, l'analyse et le développement de logiciel.
- La commercialisation de ses produits.
- Le déploiement et l'intégration des solutions au sein des systèmes informatiques des clients.

Skyrecon négocie actuellement avec des sociétés de vente indirect : distributeur, grossiste, infogérance, de manière à lui permettre de se recentrer sur sa spécialité initiale qui est le développement de solutions de sécurité.

3.4 Le positionnement du stage dans les travaux de l'entreprise

Ma tâche principale au sein de Skyrecon a été de participer à l'analyse et la conception de logiciel.

Elle a consisté, dans un premier temps, à développer une couche d'abstraction système pour permettre aux solutions Skyrecon d'être portées nativement sur plusieurs architectures (Microsoft Windows, Unix, Linux, Solaris, Mac OS X…). Ainsi qu'une couche d'algorithmes élémentaires permettant de développer plus rapidement des couches de haut niveau.

Dans un deuxième temps j'ai été chargé de réaliser un framework d'agents allégés de manière à automatiser la communication en multi-tâches des différentes sondes et modules de traitements contenus dans les solutions Skyrecon.

Je devais assurer la formation des équipes de développement ainsi que le maintient des documentations de ces technologies. (Un exemple de documentation interne est disponible dans l'annexe 8.5)

Après quoi, ma tâche à été de développer au sein de la solution WiViser un firewall réseau, une sonde WiFi, et un agent serveur de communications réseau pour la centralisation de l'administration du produit.

Dans un autre registre, j'ai été charger de présenter des démonstrations du produit aux investisseurs, distributeurs, clients et partenaires potentiels.

Ainsi que des campagnes de sensibilisation en matière de sécurité au sein des directeurs de service informatique de grandes sociétés, pour promouvoir l'essor de la technologie WiViser.

4 Travail effectué

4.1 Le cahier des charges

4.1.1 But Général

Skyrecon System a été créée de manière à répondre aux problématiques modernes en matière de sécurité informatique.

Cette décision a été prise suite à un constat simple : les techniques de défenses traditionnelles, malgré le fait qu'elles soient indispensables, ne sont plus suffisantes pour répondre aux besoins actuels des entreprises face aux nouvelles menaces ciblant leurs systèmes d'information.

En effet les attaques informatiques évoluent et sont de plus en plus complexes. Internet aide la propagation de virus, la formation accélérée des jeunes Hackers grâce aux communautés, aux forums de discutions et à la publication systématique du code source des derniers virus à la mode.

Internet n'est pas seulement un moyen de propagation des attaques et de formation des Hackers, on y trouve également grâce au peer-to-peer le code source volé de logiciels commerciaux, permettant ainsi de trouver plus aisément leurs failles de sécurité.

Internet n'est pas le seul responsable de l'évolution des attaques contre les systèmes d'information des entreprises, en effet, les technologies informatiques ont beaucoup évoluées, notamment dans le domaine de la mobilité, ce qui porte un coup de grâce au périmètre de sécurité du système d'information centralisé des entreprises.

Depuis une trentaine d'années, grâce à l'évolution des technologies de communication en matière de débit et d'intégrité des échanges d'informations, les entreprises ont pu connecter leurs différents sites entre eux ainsi que les systèmes de leurs fournisseurs et collaborateurs. Déjà les systèmes de défense comme les VPN permettaient d'authentifier les utilisateurs et de crypter les communications.

La faible interconnexion des systèmes et les limitations des moyens d'échange avec le grand public atténuaient la menace des virus informatiques.

Depuis la fin des années 80, les entreprises sont connectées directement sur Internet, hébergeant leur site, pour favoriser leurs politiques marketing, ou la vente directe. Grâce aux Intranets, elles permettent également à leurs collaborateurs de travailler à distance, et d'avoir accès au système d'information. C'est l'avènement des firewall réseau qui assureront aux entreprises la défense de leurs serveurs contre des attaques ciblées.

Déjà la couverture du parc informatique s'étend et les firewall réseau ne protègent que les serveurs situés dans les locaux de l'entreprise. Les vers informatiques qui permettaient déjà d'infecter les machines des collaborateurs, infectent par les canaux VPN ouvert, le système d'information de l'entreprise.

C'est l'apparition des IDS (Intrusion Detection System) qui, grâce à l'analyse comportementale des utilisateurs, détectent les intrus.

Les Antivirus personnels se démocratisent suite à la simplification des échanges de données. Grâce à leur système de signatures, ils permettent d'éradiquer les virus, vers et chevaux de Troie connus.

Ils seront complétés par l'offre des firewall applicatif, filtrant les connections réseaux des machines au niveau des logiciels, empêchant ainsi, les chevaux de trois de fonctionner et les vers de se répandre.

Vers la fin des années 90, c'est l'éclatement total de la couverture du parc informatique grâce aux évolutions des technologies mobiles, notamment l'augmentation de la puissance et le faible coup des ordinateurs portables, de la démocratisation des PDA, et surtout des systèmes de communication sans fil comme le BlueTooth et le WiFi.

Les attaques réseau sur les protocoles de communication se multiplient (man in the middle) favorisant le vol de données, mais également les attaques DOS sur les point d'accès WiFi, pour empêcher la communication.

Des boîtiers de surveillance ont vu le jour pour empêcher ces attaques grâce à des systèmes IDS réseau par signatures.

La multiplication des vecteurs d'attaque possible rend les solutions de sécurité de plus en plus complexes à administrer du fait de leur grand nombre et de leur diversité.

Les principaux systèmes de défense type IDS WiFi et firewall réseau sont souvent distribués sous forme de boîtiers hardware, ces types de systèmes utilisent souvent des technologies propriétaires, ce qui rend leur mise à jour difficile, et un coup d'achat et de déploiement important.

4.1.2 Explication détaillée des résultats à obtenir

C'est la raison pour laquelle Skyrecon développe WiViser, une solution logiciel de sécurité intégrée et administrable de manière centralisées, intégrant un firewall réseau, un firewall système et un IPS (Intrusion Prevention System) réseau filaire et WiFi.

Une solution embarquée totalement logiciel permet de protéger les postes clients même lorsqu'ils sont à l'extérieur du système de défense de l'entreprise. C'est une fonctionnalité primordiale pour se prémunir contre les attaques modernes.

WiViser compte résoudre également un problème récurent dans la sécurité informatique : le « zero day ». En effet, les grandes entreprises ne peuvent pas se permettre de perdre l'accès momentané à leur système d'information sans subir des pertes de rendement pouvant atteindre des sommes colossales. Généralement tous les moyens matériels sont mis à disposition pour éviter toute panne physique grâce à d'importants systèmes de redondance. Coté logiciel, cette qualité de service n'existe pas. En cas d'attaque ou de détérioration du système d'information, le seul recours est de réinstaller la dernière sauvegarde. Le problème majeur des solutions de sécurité est la réactivité : en effet les IDS se contentent de prévenir des intrusions, souvent bien trop tard, alors qu'il faudrait les bloquer, c'est ce que tentent de résoudre les IPS, de la même manière les IDS réseau et les anti-virus se basent sur des signatures pour détecter les attaques. La signature d'une attaque n'étant connue qu'après le recensement de l'infection par les éditeurs, il est déjà trop tard. Ces technologies ne sont valables que sur des attaques répertoriées et connues, c'est à l'opposé des politiques de sécurité voulant faire face à l'intrusion d'un hacker isolé. La multiplication des variantes d'un même virus ne fait qu'aggraver le problème du manque de signature.

La solution apportée par WiViser à ces problèmes est l'analyse comportementale d'applications et la corrélation des évènements collectés par les sondes déployées sur les machines clientes. La réactivité du produit est représentée par le fort couplage des sondes et des firewall présents dans la solution.

L'intégration et la centralisation de l'administration permettent une utilisation plus simple, assurant ainsi à l'administrateur la mise en place de sa politique de sécurité de façon aisée. Pour éviter une surcharge de message d'alerte dut au grand nombre de ressources sondées, un système d'apprentissage supervisé permet de filtrer les fausses alertes au niveau de la console d'administration.

Mon arrivée à Skyrecon a coïncidé avec le démarrage du développement de la solution WiViser.

4.2 Compte rendu d'activité

4.2.1 Axes d'étude et de recherche choisis

Les choix technologiques ont été fortement orientés par la demande du marché en matière de sécurité, ainsi que les contraintes technologiques qui en découlent.

Une grande partie du logiciel est écrit en C de manière à intégrer du code au noyau des systèmes d'exploitation, mais également pour permettre un portage de la solution sur des systèmes embarqués comme les Poquet PC.

Malgré le fait que le C soit un langage procédural, un souci de programmation orienté objet a été respecté de manière à rendre le code le plus stable et lisible possible.

Nous avons choisis les outils GNU présents sur toutes les architectures ainsi que le portage Windows de cette technologie : MinGW.

Un environnement de compilation à été créé grâce à un système de script Shell et de Makefile interdépendant de manière à abstraire la plateforme ciblé ainsi que le type de binaire généré (Release ou Debug). La totalité de l'architecture de développement est archivée sur un système de vesionning : SVN. Sous Windows, nous utilisons un client graphique : Tortoise SVN. Ce qui nous permet de développer à plusieurs sur le même projet et de gérer un système de droit d'accès sur les sources.

L'interface graphique quand à elle, pour des questions de rapidité de souplesse de développement et de son orientation multi architecturelle, à été développée en C# grâce au Microsoft Visual Studio .NET.

En attendant les portages Unix de cette technologie comme Mono, une interface Java allégée a été développée.

Le code ne devant pas être dépendant d'une architecture, j'ai été chargé de développer une couche d'abstraction en C permettant de cacher l'utilisation des appels systèmes dans les couches basses du logiciel. Pour simplifier le portage de l'architecture multi-thread sous Unix, l'utilisation de la librairie Pthread fut d'une grande aide.

Pour accélérer le développement de la communication sécurisée, nous avons pu utiliser la librairie OpenSSL grâce à son système de licence avantageux.

Pour la sérialisation des différents types de données, nous avons utilisé la librairie LibXml2, que nous avons abandonnée par la suite pour des raisons de fuites de mémoire et de stabilité. J'ai été chargé de réécrire un interpréteur XML pour la remplacer.

Le code C a été stabilisé grâce aux outils de désassemblage et de débogage GNU. Pour éviter toutes fuites de mémoire ou buffer overflow nous avons également utilisé Rational Purify qui permet également de détecter les erreurs de chaînes de formats lors de l'utilisation des fonctions de conversion ASCII de la libC.

4.2.2 Déroulement concret des études

4.2.2.1 Les API

Dans un premier temps, il a fallut développer plusieurs couches d'abstraction pour les différentes fonctionnalités fournies par les systèmes d'exploitation de manière à les utiliser de façon transparente dans le reste du logiciel.

Les fonctionnalités sont : le mappage de fichier en mémoire, l'exploration d'arborescence d'un système de fichier, le chargement de librairie dynamique, la gestion des accès concurrentiels aux ressources (mutex), la communication réseau, la gestion des thread, et leurs synchronisations, et enfin la gestion du temps.

Toutes ces fonctionnalités sont fournies par les systèmes d'exploitation de différente manière, et ne sont pas abstraites par la libC (Librairie de fonctionnalités standards fournies par le langage C)

Une fois l'abstraction effectuée, le code du produit est à présent commun à toutes les architectures. Il faut dorénavant écrire les différentes parties d'algorithme commun qu'on peut trouver dans les langages objets qui font cruellement défaut en C :

Un système de liste chaînée dynamique, permettant la recherche, le tri, le stockage et la destruction automatique de contenu polymorphe, mais également gérant la gestion des files, des piles et des itérateurs.

Un système de table de hash à également été développé pour permettre le stockage dynamique de données polymorphes indexées, assurant l'unicité des clef et l'accès rapide aux données.

Grâce à ces API élémentaires, des API plus complexes ont put être élaborées comme un spooler, permettant une gestion managée de thread exécutant simultanément une pile de tâches ajoutées en temps réel de façon asynchrone. Un Scheduler, permettant l'exécution de tâches asynchrones à intervalle ou heure fixe, un interpréteur / générateur de donnée XML. Une API de debug interceptant les fonctions d'allocation pour repérer les fuites de mémoire.

L'équipe Intelligence Artificielle à également écrit des API, notamment, un système expert, et un algorithme de multi pattern matching.

Certaines parties de logiciel utilisent du code spécifique, notamment dans la partie communication avec la base de données. En effet le standard de communication ODBC de Microsoft n'est pas porté pour les systèmes Unix de manière gratuite. La plupart des SGBD n'ont pas de licence d'utilisation libre ou d'autorisation d'intégration dans une solution propriétaire, ce qui empêche l'utilisation de SGBD multi plateforme comme MySQL.

Les SGBD gratuits comme MSSQL existent sous les systèmes Windows

Les SGBD Unix libre comme Postgre SQL, n'ont pas de connecteur ODBC, ce qui a entraîné l'écriture de code spécifique en fonction de l'architecture utilisée.

Pour des raisons de temps, ces parties ne sont pas encore encapsulées sous la forme d'API, mais cette étape fait partie de la prochaine phase de développement.

Les API ont été documentées grâce à Doxygen, cet outil permet d'extraire les commentaires des entêtes des fonctions pour générer de la documentation au format WEB, permettant aux équipes de développement d'utiliser les API le plus tôt possible.

Pour tester les API et pour compléter les documentations, les sources des jeux d'essais sont rendues publiques pour servir d'exemples.

Maintenant que les bases de l'architecture de développement des solutions Skyrecon sont posées, on peut démarrer le développement d'application proprement dit.

4.2.2.2 Le Framework

Tout d'abord j'ai été chargé de développer le noyau commun des logiciels Skyrecon, que nous avons appelé par abus de langage : le framework.

Ecrit avec les API d'abstraction vu précédemment, il est multi-plateforme et doit être à usage multiple, il ne doit pas être spécifique à WiViser.

En effet il est également utilisé dans la technologie DataViser en cours de développement à l'heure actuelle, son code évolue de manière à apporter toutes les fonctionnalités nécessaires aux différents logiciels dont il sert de noyau tout en respectant la rétro compatibilité.

Le framework est le point d'entrée des logiciels Skyrecon, il régit le chargement dynamique des agents et la communication entre les agents sondes et les agents de traitements de chaque solution. Il permet une exécution des agents en multi-tâches, manage les tâches de traitements, et permet une communication inter agents, synchrone ou asynchrone.

Grâce à un système de script XML il permet également de maintenir des variables d'environnement communes ainsi que de lancer des fonctionnalités exposées par les agents.

La notion d'agent à été utilisé car, dans les premières versions du cahier des charges, le framework devait être développé par l'équipe Intelligence artificielle et devait reposer sur les spécifications CORBA en matière d'architecture distribuée.

Pour des problèmes de temps et de contraintes techniques, les objectifs ont été revus en simplifiant la notion d'agent au minimum de manière à avoir les fonctionnalités élémentaires et un moteur agent le plus rapidement possible.

Certaines fonctionnalités du modèle CORBA jugées dans un premier temps inutiles dans les solutions Skyrecon, ont donc été amputées. Ont été supprimées :

- La notion d'agents mobiles, simplifiant le moteur n'ayant plus à interpréter le code des agents en fonction de leur plateforme d'origine.
- L'exposition des types de données échangées dans les messages permettant l'abstraction de la communication réseau inter-agents par la sérialisation automatique des données transmises.

Les agents sont donc des librairies dynamiques implémentant des systèmes de construction hérités de la programmation objet qui peuvent, grâce au framework, exposer leurs points d'entrées, permettant la réception de messages et leurs interactions mutuelles.

4.2.2.3 Architecture de WiViser

WiViser est un logiciel modulaire découpé en plusieurs grandes parties.

Tout d'abord la sonde est installée sur les postes clients à sécuriser. Cette partie est nommé WiViser Client.

Puis le système de déploiement des politiques de sécurité, centralise les alertes et messages remontés par les sondes dans une base de données. Ce système se trouve sur une ou plusieurs machines serveur. Cette partie est nommé WiViser Pilot.

Puis la dernière partie : la WiViser Console, qui est l'interface graphique du logiciel permettant à l'administrateur du site de définir ses politiques de sécurité et de consulter les attaques détectées par les sondes et leur statistiques correspondantes.

Le WiViser Pilot et le WiViser Client sont architecturés autour du framework décrit précédemment. L'interface elle, est réalisée en C#, chaque partie de la solution communique avec les autres grâce au système de communication sécurisé proposé par les API sous-jacentes.

Grâce au système d'agent du framework, toutes les fonctionnalités du WiViser Client et du WiViser Pilot sont indépendantes, autonomes et interchangeables à volonté.

Cet avantage a permit de segmenter le développement du logiciel et de répartir plus facilement les tâches au travers des équipes de développement en contournant les problèmes de version des binaires et d'accès concurrentiels au code source.

4.2.2.4 Les Agents

Pour la solution WiViser, j'ai été chargé de développer plusieurs agents.

L'agent de traitement modprint. Cet agent est commun au WiViser Pilot et au WiViser Client. Son rôle est d'afficher les messages d'information et les messages d'erreur envoyés de façon synchrone ou asynchrone par les autres agents, pour cela il fallu gérer l'accès concurrentielle aux ressources d'affichage. Son autre but est d'archiver et de dater ces messages lorsque le logiciel fonctionne en arrière plan de façon invisible, ce qui est systématiquement le cas chez le client.

Grâce à la centralisation du système de message, tous les événements sont recensés dans des fichiers de log, et pourront être routés, dans un deuxième temps, dans une base de données.

L'agent de traitement modserver est un agent écrit pour le WiViser Pilot. Il a été créé dans le but de régir les connexions réseau sécurisées des machines clientes, d'assurer que leurs configurations sont à jour, et de la mettre à jour leur politique de sécurité dans le cas contraire.

L'agent modserver communique avec l'agent de traitement modalert écrit par Monsieur Fayçal Daira, qui permet aux alertes d'êtres corrélées par l'agent modexpert écrit par Monsieur Yann Torrent et archivées dans la base de données, de manière à les rendre consultable par l'interface graphique.

L'agent modserver communique également avec l'agent modmulti développé par Monsieur Jean Baptiste Lernout, à chaque connexion et déconnexion de client. Ceci permet de gérer les clients en fonction de la licence du produit. Cet agent permet également la répartition de charge dans le cas ou le site propose plusieurs serveurs WiViser Pilot.

Grâce à un protocole de communication commun et ouvert, Monsieur Loïc Dupuis a pu écrire l'agent sonde modclient correspondant sur le WiViser Client, permettant de recevoir la configuration et d'envoyer les messages et les alertes.

La configuration du Client ou du Pilot est déployée pour chaque agent par un agent commun modconf également écrit par Monsieur Loïc Dupuis. Il a également développé l'agent modgui permettant de recevoir les politiques de sécurité de l'interface graphique et de les mettre à disposition des sondes sur le serveur.

Coté sonde j'ai été chargé d'écrire un agent firewall nommé Thor. Cet agent est un firewall réseau dynamique qui a pour but d'empêcher l'utilisation de certains protocoles en fonction de la politique de sécurité du site. Grâce à son orientation dynamique, le framework permet l'interaction avec d'autres agents sonde, comme l'IDS écrit par Monsieur Benjamin Villedieu. Ceci permet de le transformer en IPS. En effet, la génération de règles de filtrage à la volée, complète les règles de filtrage définies par l'administrateur, bloquant ainsi les attaques réseau dynamiquement détectées.

J'ai également été chargé de développer l'agent sonde modap, cet agent doit analyser les caractéristiques WiFi du site ou se trouve le poste client sécurisé. En fonction des politiques de sécurité de la machine, on empêchera les adaptateurs réseau de se connecter à des points d'accès interdits, inconnus ou les connexions points à points (Ad-Hoc).

La complexité de cet agent réside dans le fait qu'il est bâti sur une technologie émergeante utilisant des API système mal documentées ou non terminées, en perpétuelle évolution. Dans les première version de l'agent, une partie résidant dans le noyau du système d'exploitation était nécessaire, puis finalement remplacée par un agent entièrement bâti en environnement utilisateur grâce à l'apparition de nouveaux appels systèmes. La stabilité de l'agent a dut être retravaillé pour la compatibilité sur les différentes versions des systèmes d'exploitation Microsoft ne mettant pas encore à disposition ces différentes API.

Pour empêcher les connexions, l'agent modap génère des règles de filtrage envoyées grâce au framework à l'agent Thor.

4.2.2.5 Architecture du firewall réseau Thor

Le firewall réseau Thor est un filtre qui permet de détruire les paquets de données indésirables arrivant sur les adaptateurs réseau de la machine protégée. Sous les systèmes Unix, les iptable permettent nativement de filtrer les paquets, sous les systèmes Microsoft précèdent le service pack 2 de Windows XP, aucun filtre n'était fourni. C'est pourquoi il a fallut réécrire un firewall dans le noyau de Windows. (CF Annexe 8.1 et 8.2)

Ces filtres écrits à l'intérieur du système permettent de détruire les attaques réseau avant même qu'ils atteignent les services installés sur le poste sécurisé, ceci permet de les protéger contre les exploits permettant aux chevaux de Troie de s'installer sur la machine. (CF Annexe 8.4)

Ecrire un pilote dans le noyau du système rajoute de la complexité car les outils fournis par la libC en mode utilisateur ne sont plus disponibles, ainsi que la majorité des interfaces utilisateurs simplifiant l'utilisation des appels systèmes.

Il a également fallut développer une interface de communication permettant au WiViser Pilot de communiquer avec la partie driver présente dans le noyau du système.

Pour cela, une api a été créée de manière à encapsuler la gestion d'événement et la gestion d'échange de données mis en place par le système de contrôle d'entrée / sortie pour pilote communément appelé IOCTL.

Le noyau Windows fourni tout de même une api permettant d'écrire des composants réseau sans se soucier du nombre et du type de matériel installé cela pour les différentes versions des systèmes Microsoft. Cette interface s'appelle NDIS (Network Device Interface System)

Le firewall Thor est donc composé de deux parties physiques :

- une partie utilisateur représentée par un agent pour le framework du WiViser Client.
- une partie système représentée par un pilote NDIS.

4.2.2.6 L'agent Thor

L'agent Thor a un usage multiple :

Grâce au framework, il permet aux autres agents d'envoyer des règles de filtrage dynamiquement au pilote grâce à l'api de communication introduite précédemment.

Il gère la durée de vie des règles dynamiquement générées.
La durée de vie limitée des règles dynamiques permet au poste de se reconnecter à la machine ayant commis l'attaque. Lors d'une attaque isolé, l'utilisateur peut continuer sont travail comme s'il ne s'était rien passé. De ce fait le blocage permanant du système d'information ou l'isolation de la machine sont évités.

Une API de génération de règles de filtrage a également été réalisée de manière à centraliser et homogénéiser le code des agents générant ces règles. Cette API a été intégralement écrite en macro de manière à ce que le pré processeur C génère le code le plus rapide possible, car cette api est utilisée dans le pilote NDIS pour comparer les règles aux paquets.

Chaque règle de filtrage est une combinaison d'ensembles ou de valeurs attendus dans un paquet autorisé ou néfaste. (CF Annexe 8.5 et 8.6)
Si le paquet est néfaste, c'est-à-dire qu'il contient par exemple un protocole de communication non autorisé par la politique du site, la règle indiquera au firewall que le paquet doit être détruit. Si par contre le paquet est autorisé, par exemple un paquet dont l'adresse de la machine émettrice appartient à l'ensemble des machines du site, alors la règle indiquera au firewall, que le paquet est accepté.
(CF Annexe 8.7)

Les règles de filtrage par défaut sont émises par l'administrateur grâce à la politique de sécurité qu'il à mise en place. Elles sont stockées dans le fichier de configuration du WiViser Client sous le format XML.
Un compilateur de règles de filtrage à été écrit de manière à transformer les règles de filtrage XML en règles de filtrage binaire interprétables par le pilote.

Ce compilateur contient un analyseur syntaxique spécifique complétant celui de l'analyseur syntaxique XML permettant de détecter les erreurs de syntaxe ainsi que les ambiguïtés qui peuvent en découler. Ce système permet une première étape d'optimisation des règles qui consiste à renseigner les protocoles sous-jacents

définis par le protocole explicitement filtré par la règle. Ceci permet une comparaison plus rapide des paquets.

Une dernière API fut développée pour permettre à l'agent Thor de faire une optimisation en profondeur des règles.

En effet le système de filtrage d'un firewall correspond à une liste de règles triées par priorité. Chaque règle contient des valeurs ou intervalles de valeurs qui permettront de savoir si les paquets correspondent ou non à la règle. Pour chaque paquet reçut ou envoyé, on vérifie dans l'ordre de priorité si chaque règle répond au critère du paquet. Si cela n'est pas le cas, on passe à la règle de priorité inférieure, si c'est le cas, la règle détermine si le paquet doit passer ou doit être détruit.

On peut se rendre compte que plus la politique de sécurité d'un site est sévère, plus le nombre de règle peut grandir. Dans une politique de « black list », c'est-à-dire ou l'on spécifie seulement les critères des paquets indésirables, la majorité des paquets transitant sur le réseau ne correspondent à aucune règle. Donc chaque paquet est testé avec l'intégralité des règles sans succès d'interprétation. Ce cas là étant rare mais problématique pour des raisons de performance, l'optimisation en profondeur des règles était nécessaire.

L'optimisation en profondeur consiste à vérifier que les règles de priorité inférieure ne sont pas recouvertes par les ensembles de critères de filtrage définis par les règles de priorité supérieure.

Ce système permet de synthétiser la liste de règles à envoyer au pilote, ce qui accélère le temps de traitement lorsque d'importants flux de données sont transmis.

4.2.2.7 Développement de Thor dans le noyau de Windows

La partie pilote a également été développée en plusieurs parties. Dans un premier temps pour répondre au besoin de segmentation lié au développement des api vu précédemment, mais également par le fait que le noyau Windows n'est pas monolithique.

De longues heures de test et de développement ont été nécessaires pour stabiliser le code du pilote, dû à certaines contraintes de développement mal connues liées à la programmation noyau.

Tout d'abord le multi-tâche est omniprésent dans le noyau Microsoft.
Les systèmes de synchronisation classique existent pour éviter ce genre de problème, mais le système d'exploitation fonctionne sur plusieurs niveaux de priorités, ces priorités sont appelées IOPL. Plus les threads noyaux sont lancés avec une priorité haute, plus leur temps d'exécution est rapide, et plus ils doivent rendre la main rapidement.
Les threads de haute priorité sont généralement lancés par des interruptions matérielles et ne peuvent pas être interrompus par les threads de priorité inférieure. On peut donc constater que le système de synchronisation adéquat est bien plus difficile à mettre en place pour les programmes proches des périphériques.

Généralement les tâches lourdes exécutées par les threads de haute priorité doivent être déléguées à des threads de priorité inférieure par des mécanismes de procédure asynchrones.

D'autres contraintes existent dans la programmation noyau :
L'accès aux api du noyau n'est pas possible dans toutes les priorités, en effet dans la documentation du Driver Development Kit de Microsoft (DDK) le niveau de priorité d'utilisation de chaque fonction est spécifié, par exemple dans la plupart des cas, l'utilisation d'une opération bloquante est interdite dans les tâches de haute priorité.
Ces problèmes ne sont pas les plus graves, en effet la gestion de la mémoire dans le noyau est bien plus complexe. La pile est limitée à 12 kilos octets par thread !

La mémoire allouée n'est pas forcément accessible en priorité haute, en effet par défaut, la mémoire est allouée en mode paginé, c'est-à-dire qu'elle peut être librement déplacée par Windows sur le disque dur dans le fichier d'échange du système de manière à augmenter l'espace de mémoire virtuelle utilisée par les applications les plus sollicitées. Ceci peut entraîner des erreurs de segmentation entraînant le crache de la machine. Pour empêcher ces problèmes, il faut utiliser l'espace de mémoire physique dit non paginé de manière à y stocker le minimum d'informations nécessaires au fonctionnement des threads de haut niveau. Cette ressource doit être utilisée avec modération car elle est physiquement limitée en taille.

Grace au système de règles de filtrage compilées, chaque règle, quelque soit sa complexité, fait 84 octets. Ceci facilite son stockage dans la mémoire non paginé ainsi que la vitesse d'accès et d'interprétation.

4.2.2.8 Intégration de Thor en tant que pilote intermédiaire NDIS.

NDIS permet de créer différents types de pilote. Les pilotes physiques de carte réseau et les périphériques virtuels sont décrits sous le terme générique de miniport. Les services permettant entre autre le routage des paquets, la qualité de service, les systèmes de chiffrement ou d'authentification sont regroupés sous la nomination protocole. NDIS permet également la création de pilotes intermédiaires se situant entre les miniports et les protocoles, permettant ainsi d'avoir accès à tous les paquets qu'ils soient entrants, sortant, générés par une application ou un service et reçu par n'importe quel périphérique.

C'est l'emplacement idéal pour créer un système de défense contre les attaques protocolaires sur les systèmes Windows.

L'API NDIS est d'une grande aide pour abstraire le nombre d'adaptateurs réseau et de protocoles installés sur la machine, mais le développement sur cette architecture n'est pas très aisé. En effet il existe plusieurs versions de l'api NDIS. Nous fonctionnons actuellement sur la version 5.1 sous Windows XP et 5 sous Windows 2000, la version 6 est en cours de développement chez Microsoft. De gros efforts sont nécessaires pour permettre la compatibilité ascendante des pilotes développés, notamment dans la réécriture de fonctionnalités dans les points d'entrée obsolète de l'API.

La création du firewall Thor au sein de l'architecture NDIS à été inspirée de l'exemple du DDK : passthru.

Ce programme d'exemples illustre la création d'un pilote intermédiaire NDIS qui, comme son nom l'indique, est complètement transparent et laisse passer les paquets comme s'ils n'existaient pas. Grâce à lui, nous pouvons créer le squelette d'un pilote intermédiaire. Une fois que les contraintes de synchronisation et de gestion de mémoire vu précédemment auront été résolues, nous pourront implanter le pilote de filtrage Thor proprement dit.

Le firewall est donc décomposé en quatre grandes parties :

- le pilote.
- le miniport intermédiaire.
- le protocole intermédiaire.
- le filtre.

Le pilote est chargé d'initialiser et de décharger le firewall. Il enregistre le miniport intermédiaire et le protocole intermédiaire au niveau de l'api NDIS, et initialise le filtre. Il permet également de créer le périphérique virtuel et les événements de communication pour permettre l'interaction avec l'agent Thor et les différents outils de test conçus pour son élaboration.

Le filtre compare les paquets aux règles de filtrage grâce à l'API de règle développée précédemment, et détermine si chaque paquet doit être accepté ou détruit.

Le miniport intermédiaire permet d'intercepter tous les paquets sortants, son niveau de priorité est élevé car il se trouve sur les pilotes des cartes réseau, donc proche des périphériques. Il appellera le filtre pour déterminer l'action à effectuer sur les paquets émis.

Le protocole intermédiaire permet d'intercepter tous les paquets entrants. Son niveau de priorité est moins élevé car il se trouve du coté des services applicatifs, ou le temps de traitement est moins critique. Il appellera également le même filtre pour gérer les paquets.

Le filtre à un rôle complexe car c'est lui qui doit gérer l'accès concurrentiel sur les règles de filtrage en fonction du niveau de priorité de l'appelant. En effet, le pilote régissant l'interaction avec la partie applicative du firewall peut recevoir à tout moment de nouvelles règles de filtrage. De la même manière les cartes réseau modernes gérant le full duplex, c'est-à-dire la réception et l'envoi simultané de paquets peuvent appeler simultanément le protocole et le miniport intermédiaire. De plus la présence de plusieurs adaptateurs réseau entraîne l'envoi ou la réception simultanée de paquet, ce qui force le code du filtre, du miniport et du protocole intermédiaire à être réentrant. Même chose pour le pilote car son périphérique virtuel peut avoir une interaction avec plusieurs clients applicatifs, ce qui était régulièrement le cas dans le cadre des phases de test.

Pour les tests les plus avancés, Microsoft certifie la qualité des pilotes avec le label WHQL (Microsoft Windows Hardware Quality Labs) cette certification est obtenue en réalisant la batterie de test HCT (Hardware compatibility Test). Pour tester les pilotes NDIS, le HCT contient l'outil NDISTest qui nous a permis de détecter les problèmes de stabilité dans le code du firewall. Ces tests assurent également la portabilité des pilotes NDIS sur des Pocket PC fonctionnant sous Windows CE. (CF Annexe 8.3)

4.2.2.9 Promulgation de WiViser sur le marché informatique.

Mon premier rôle dans l'émergence de la solution WiViser sur le marché a été celui d'un assistant technique lors des présentations de la technologie aux investisseurs. J'ai été chargé d'assister l'équipe de commerciaux et le chef de projet aux démonstrations faites pour les sociétés de capital risque que sont Auriga Partners et Galileo Partners. Des démonstrations ont également été faites à l'ANVAR, organisme national spécialisé dans l'aide au développement des entreprises innovantes.

Dans le même axe de travail, j'ai effectué des présentations similaires pour les sociétés de distribution de logiciels ITWay Softway et Allasso. Puis dans le cadre d'un partenariat technologique avec la société Cisco Systems.

Mon travail dans ces présentations consistait à présenter le fonctionnement de la technologie par différents types d'attaques : Man In the Middle, DOS, Virus, Vers, de manière à illustrer les explications effectuées par le chef de projet et les commerciaux.

Les scénarios et jeux d'essai présentés lors des démonstrations étaient préparés à l'avance. Ils étaient accompagnés d'explications succinctes du fonctionnement du produit. Ces explications pouvaient être plus détaillées en fonction des questions des différents interlocuteurs.

Dans un deuxième temps, des responsabilités plus importantes m'ont été confiées : rédiger et présenter la technologie du produit lors d'un séminaire organisé par Skyrecon Systems. Le but de ce séminaire était de sensibiliser les responsables informatiques des grandes entreprises aux nouveaux problèmes de sécurité et de présenter la technologie WiViser au grand public.

4.3 Interprétation et critique des résultats

L'utilisation d'un langage orienté objet aurait simplifié le code et réduit les coups de développement en terme de réutilisation de l'existant. Les contraintes technologiques comme la programmation noyau ou le développement sur les systèmes embarqués en ont décidé autrement.

Le travail d'ingénieur est bien entendu de choisir judicieusement les technologies en fonction du travail à réaliser. Si les technologies les plus commodes ne sont pas disponibles pour satisfaire la demande, le rôle de l'ingénieur est également de rendre le travail réalisable avec la qualité qu'il aurait pu avoir avec des technologies plus modernes ou des outils de RAD.

Malgré le fait qu'elle se résume à travailler sur des problématiques déjà résolues, la réécriture des couches basses apporte bien des avantages, notamment d'avoir accès au code source, ce qui permet une portabilité infinie sur des architectures totalement hétérogènes.

L'importance du code en découlant impose une rigueur au niveau des tests unitaires, qui finit par s'appliquer sur l'ensemble du logiciel, ce qui donne un retour sur investissement intéressant. Il faut tout de même faire attention à ne pas sombrer dans la seule résolution de problèmes techniques de bas niveau qui ont tendance à nous éloigner de la conception initial du projet. L'analyse et la conception architecturelle du projet doivent rester la clef de voûte du logiciel. Bien que la technologie modifie la conception, la seule variable modifiant complètement l'architecture du cahier des charges ne peut être que le temps imparti à la réalisation du projet.

En effet des contraintes de temps ont du amputer certaines fonctionnalités prévues dans les versions initiales du cahier des charges. C'est également le rôle de l'ingénieur de savoir évaluer le coup des tâches en temps de développement, mais également de pouvoir retomber sur ses pieds en cas de réorientation brutale. La programmation orientée objet entraînant l'écriture d'un code extrêmement réutilisable a permis un gain de temps énorme sur les restructurations architecturelles de haut niveau.

5 Conclusion

Le stage au sein de la société Skyrecon m'a permis d'avoir une expérience épanouissante en terme d'analyse et de conception logiciel. Mon arrivée au départ de la réalisation de la solution WiViser à autorisé une grande liberté d'expression. Ce qui rend le travail agréable est très valorisant.

Cette aventure a également été très instructive notamment au niveau de l'expérience acquise en programmation noyau, mais surtout en terme de formation sur des terrains totalement nouveaux telles que commercialisation et les études marketing du monde de l'entreprise.

Les avantages offerts par le travail au sein de petites PME donnent l'accès à une grande palette de responsabilités et d'effectuer des tâches aussi diverses que variées.

Le travail en équipe a été très important, et a permis de clôturer dans la continuité ma formation au sein d'Epitech.

6 Bibliographie

Printing Communications Associates, Inc.
http://www.pcausa.com

The Microsoft Developer Network
http://msdn.microsoft.com

Windows Hardware Driver Central
http://www.microsoft.com/whdc/default.mspx

Page Man Kheops Linux
http://linux-kheops.com/doc/man/manfr/man-html-0.9/index.php

The Single UNIX ® Specification, Version 2
http://www.opengroup.org/onlinepubs/007908799/xsh/pthread.h.html

The Open Source toolkit for SSL/TLS
http://www.openssl.org

7 Glossaire

AD-HOC : Mode de communication des périphériques WiFi permettant leurs interconnections sans point d'accès.

Agent : Partie de logiciel autonome, servant de sonde ou de module de traitement dans le cas d'un Agent du Framework Skyrecon.

Allasso : Société d'édition et de distribution de logiciels.

Antivirus : Logiciel permettant la détection des virus informatiques et la désinfection des machines contaminées.

ANVAR : Agence nationale de valorisation de la recherche également appelée Agence Française de l'innovation, est un organisme sous la tutelle du Minefi et du ministère de la Recherche dont la tâche est d'aider les entreprises françaises innovantes.

AP : Access Point : Point d'accès.

API : Application Programming Interface : Morceau de logiciel permettant d'abstraire une complexité d'implémentation technique sous jacente.

Appel Système : Fonctionnalité offerte par le système d'exploitation permettant aux logiciels d'avoir accès aux périphériques de la machine.

ASCII : American Standard Code for Information Interchange : Jeu de caractère standard utilisé dans l'industrie informatique.

Auriga Partners : Société de capital risque.

BlueTooth : Système de communication sans fil, léger pour système informatique embarqué.

Buffer overflow : Problème de sécurité récurant lié à l'absence de test des tailles des blocs de mémoire dans les logiciels. Ils permettent à un Hacker de modifier le comportement des logiciels.

C : Langage de programmation.

C++ : Langage de programmation rajoutant les concepts objet et la généricité au langage C

C# : Langage de programmation multi-plateforme de Microsoft simplifiant la syntaxe du langage C++.

Cheval de Troie : Logiciel indésirable ouvrant une brèche sur une machine pour permettre à un Hacker d'y accéder.

Cisco Systems : Société spécialisé dans le réseau informatique

Compilation : Etape permettant à un fichier source d'un logiciel de se transformer en fichier binaire interprétable par le processeur d'un ordinateur.

CORBA : Common Object Request Broker Architecture : Modèle d'architecture pour le développement objet dans les environnements hétérogènes distribués.

Débogueur : Logiciel permettant de tracer l'exécution d'un programme, permettant ainsi d'en trouver les erreurs de fonctionnement.

Désassembleur : Logiciel permettant de visualiser le contenu d'un fichier binaire exécutable.

DataViser : Solution de sauvegarde automatique proposée par Skyrecon Systems.

DDK : Driver Devellopement Kit : Environnement de développement de Microsoft pour écrire des pilotes Windows.

DOS : Deny of Service : Attaque (exploit) visant une faille de sécurité d'un service (le plus souvent réseau) permettant à un individu mal intentionné de s'introduire dans le système d'information.

Doxygen : Outil de génération automatique de documentation de code source.

Exploit : Nom donné par les Hackers à leurs attaques sur les failles de sécurité des logiciels.

Firewall : Logiciel permettant l'interdiction d'utilisation ou le filtrage de certaine ressource.

Firewall applicatif : Firewall permettant d'interdir à des logiciels l'accès au réseau.

Firewall réseau : Firewall permettant le filtrage des communications réseau en fonction de leur contenu.

Firewall système : Notion introduite par Skyrecon pour représenter un Firewall chargé de filtrer l'utilisation des appels systèmes.

Framework : Noyau des applications Skyrecon Systems permettant la gestion des agents.

Full duplex : Propriété indiquant la possibilité d'émettre et de recevoir des informations simultanément.

Galileo Partners : Société de capital risque.

GNU : GNU's Not Unix : Appellation générique des logiciels libres écrits par la Free Software Foundation.

Hacker : Délinquant informatique.

Hash : Valeur résiduelle obtenue par calcul, permettant une association avec une valeur plus complexe.

HCT : (Hardware compatibility Test) Batterie de tests fournie par Microsoft pour obtenir le label WHQL.

IDS : Intrusion Detection System : Logiciel permettant la détection d'intrusion.

Internet : Réseau commun mondiale.

Intranet : Réseau interne aux entreprises utilisant les technologies de communication fournie par Internet.

IOCTL : (Input Output ConTroL) Système de contrôle d'entrée / sortie fourni par les systèmes d'exploitation pour permettre la communication entre les applications et les pilotes.

IOPL : (Input Output PriviLege) Propriété identifiant les privilèges des tâches dans le noyau de Microsoft Windows.

IPS : Intrusion Prevention System : logiciel permettant, en plus de détecter les intrusions, de réagir pour empêcher ces dernières.

Iptable : Mécanisme de filtrage des paquets réseau fourni dans les systèmes Unix.

ITWay : société de distribution de logiciel.

LIBC : librairie C : Librairie de fonctionnalités standards fournie par le langage C.

Librairie dynamique : Morceau de logiciel dynamiquement chargeable et échangeable par ce dernier.

LINUX : Système d'exploitation libre basé sur les systèmes UNIX.

MAC OS-X : Système d'exploitation d'Apple.

Makefile : Scripte permettant d'automatiser la compilation des gros logiciels.

Man In The Middle : Technique d'usurpation d'identité sur les réseaux informatiques.

Mémoire Paginé : Mécanisme de mémoire virtuelle volatile fournis par les systèmes d'exploitation regroupant la mémoire RAM et un fichier d'échange virtuel.

Mémoire Non Paginé : Mécanisme fourni par les systèmes d'exploitation pour accéder à la mémoire RAM physique présente sur la machine.

Minefi : Ministère de l'économie des finances et de l'industrie.

MinGW : portage pour Microsoft Windows des logiciels de compilation GNU.

Miniport : Nomination générique des pilotes NDIS regroupant les pilotes physiques de carte réseau et les périphériques virtuels.

MSSQL : SGBD gratuit fournis par Microsoft fonctionnant sur les systèmes Windows.

Multi pattern matching : Algorithme permettant une comparaison rapide d'une signature dans un block de donnée.

Multi-thread : Multi-tâche : appellation générique indiquant la possibilité à un logiciel d'effectuer plusieurs tâches simultanément.

Mutex : MUTual EXclusion : Appellation générique d'un mécanisme informatique permettant l'accès concurrentiel à une ressource.

MySQL : SGDB à usage privé libre, ce qui l'a rendu très utilisé dans le monde du WEB.

NDIS : Network Device Interface System : architecture de développement proposé par Microsoft pour écrire des pilotes réseau.

NDISTest : Logiciel de test spécialisé dans le NDIS contenu dans le HTC de Microsoft.

ODBC : Open DataBase Connectivity : Standard de communication avec les bases de données définie par Microsoft.

Open Source : Qualité d'un logiciel dont le code source est publié.

OpenSSL : Librairie permettant la communication sécurisée entre logiciel.

Paquet : Nom générique donné aux blocs d'information circulant sur le réseau.

PDA : Personal Data Assistant : Ordinateur miniature transportable permettant le stockage d'agendas et d'autres informations légères comme des plans et des notes…

Peer-to-peer : Système répandu d'échanges de données entre les utilisateurs d'Internet. Son point fort est qu'il n'a pas besoin de serveur centralisé pour fonctionner.

Périphérique virtuel : Système permettant à une application de s'attacher à un pilote pour communiquer grâce à un fichier virtuel (Voir IOCTL)

Pilote : (Driver) : nom donné à un logiciel se trouvant dans le noyau du système d'exploitation.

Pilote intermédiaire : pilote NDIS permettant d'intercepter les communications réseaux.

Pocket PC : Voir PDA

Point d'accès : Terme utilisé dans le WiFI pour identifier un périphérique réseau susceptible d'accepter des communications sans fil et éventuellement de les router sur un réseau filaire. Ce périphérique est également appelé borne.

PostGreSQL : SGBD open source libre fonctionnant sur les systèmes Unix.

Pré processeur : Outil permettant de générer automatiquement du code répétitif. Il intervient généralement avant la compilation du code source.

Pthread : Librairie permettant l'écriture de logiciel multi-tâches sous les systèmes Unix.

RAD : Rapid Application Development : Outil de développement rapide pour créer des applications usuelles.

RAM : (Random Access Memory) Composant électronique contenu dans les systèmes informatiques, permettant le stockage momentané d'informations volatiles à grande vitesse.

Rational Purify : Logiciel permettant de tester la qualité d'autres logiciels.

Réentrant : capacité d'un algorithme à être exécuté plusieurs fois de façon simultanée.

Scheduler : Planificateur de tâche.

Script Shell : Langage de programmation allégé permettant aux administrateurs d'automatiser les tâches effectuées par les outils des systèmes d'exploitation.

Service pack : Nom donné par Microsoft pour identifier les mise à jour ou correctif de ses systèmes d'exploitation.

SGBD : Système de Gestion de Bases de Données.

Skyrecon Systems : éditeur de logiciel de sécurité.

Softway : Société de distribution de logiciel.

Solaris : Système d'exploitation de Sun Micro system.

Sonde cliente : voir WiViser Client.

Spooler : Système permettant la gestion d'exécution de tâches simultanées.

SVN : Sub VersioN : Logiciel spécialisé dans l'archivage de fichier source et le travail de groupe sur l'édition de logiciels.

Système de fichier : Architecture arborescente permettant à un système d'exploitation de stocker des données.

Système expert : Système d'intelligence artificiel permettant de résoudre des faits grâce à un moteur de règles.

Thor : nom de code du firewall réseau de la solution WiViser. (Dieu du tonnerre dans la mythologie scandinave)

Thread : Concept représentant une tâche à effectuer par un logiciel.

Tortoise SVN : Client graphique Windows pour le logiciel SVN.

UNIX : Système d'exploitation standardisé multi-plateforme.

Ver : (Informatique) Virus informatique ayant la capacité de se propager par le réseau.

Virus : (Informatique) logiciel néfaste ayant la capacité de se multiplier et d'infecter les logiciels d'une machine.

VPN : Virtual Private Network : Système permettant le cryptage des communications réseaux.

WEB : Terme générique identifiant la toile de sites Internet.

WHQL : (Microsoft Windows Hardware Quality Labs) Label de qualité des pilotes fournis par Microsoft.

WiFi : WIreless FIdelity : technologie permettant la communication réseau sans fil.

Windows : Système d'exploitation de Microsoft.

WiViser : Solution de protection des postes clients proposée par Skyrecon System.

WiViser Client : Partie cliente de la solution WiViser, également appelée sonde.

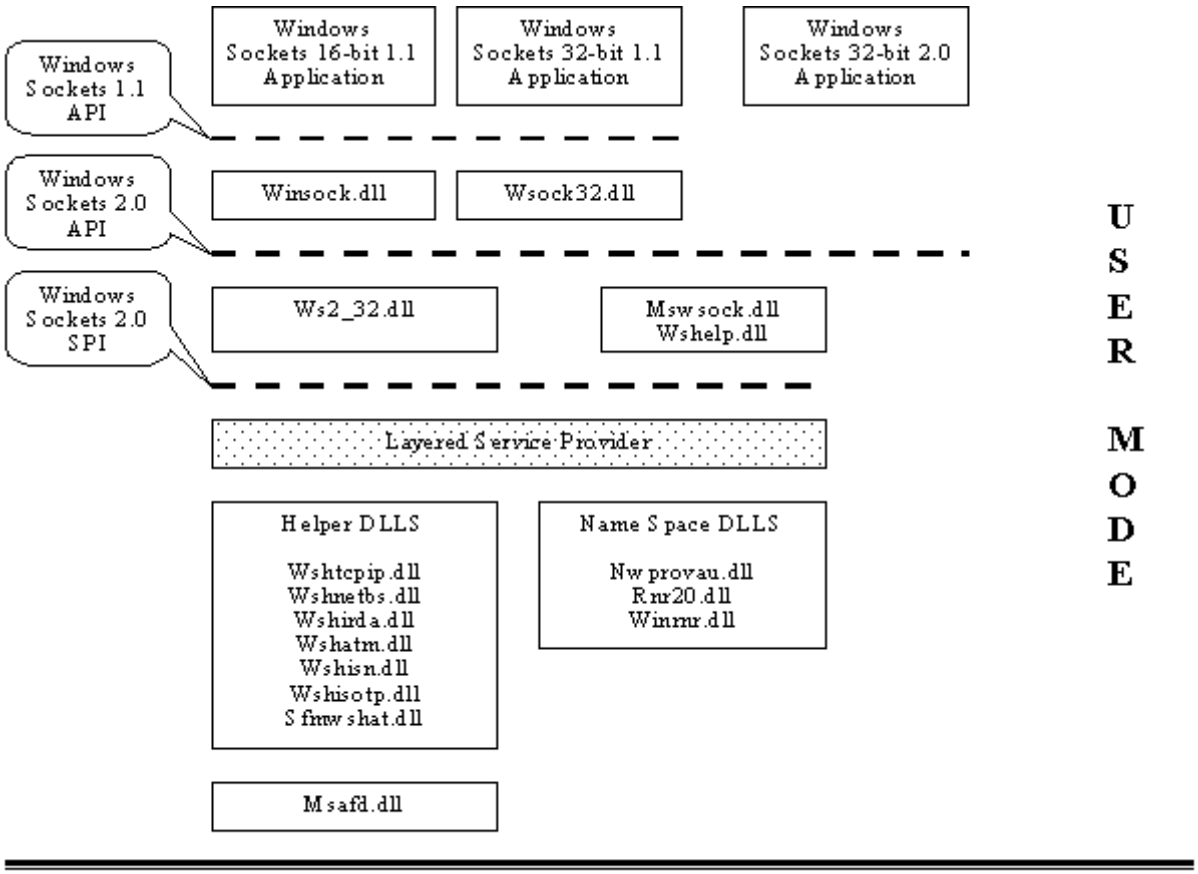
WiViser Console : Interface graphique permettant l'administration centralisée de la solution WiViser.

WiViser Pilot : Partie serveur de la solution WiViser.

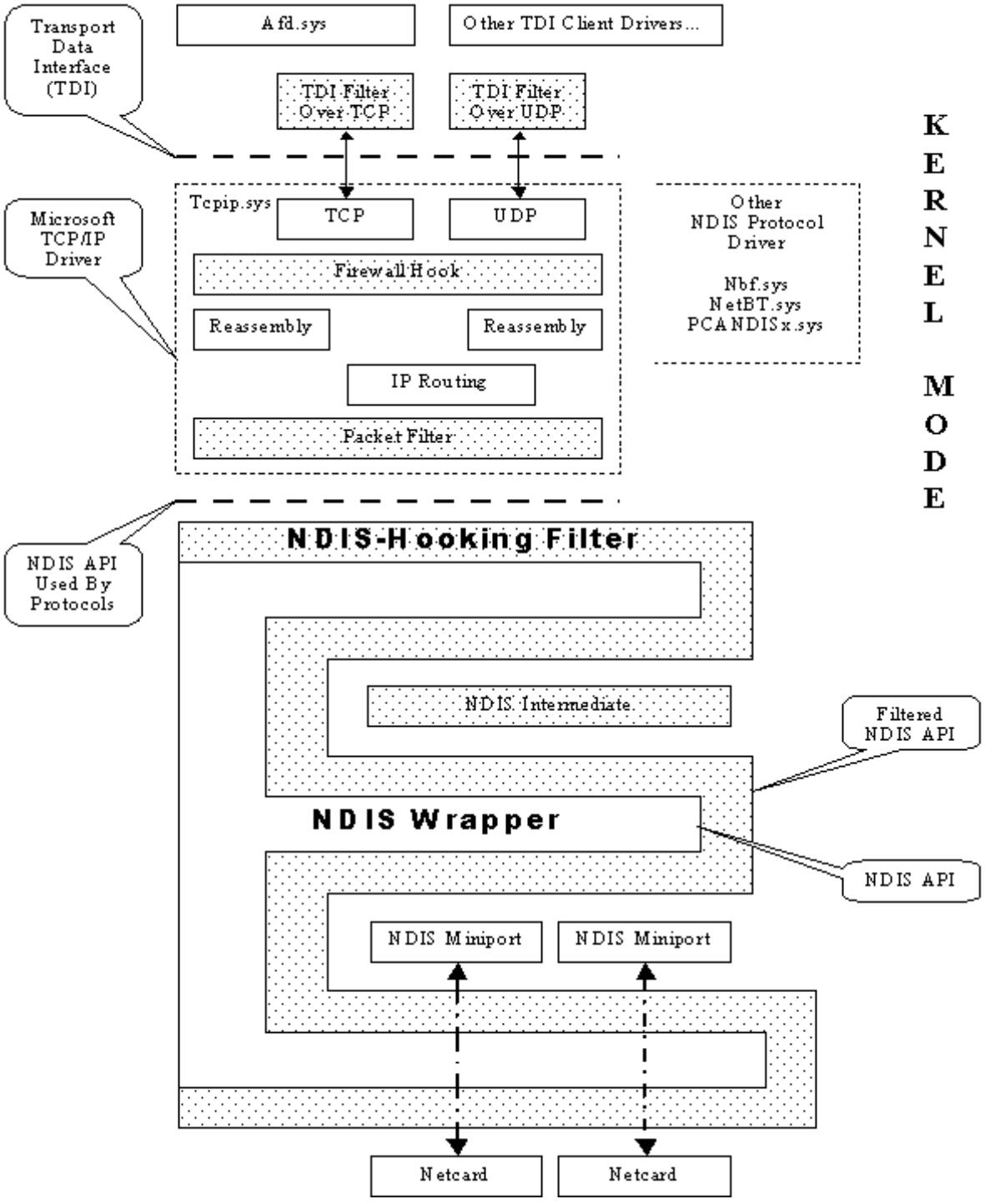
XML : eXtensible Markup Language : format de données standardisées pour l'échange d'informations.

8 Annexe

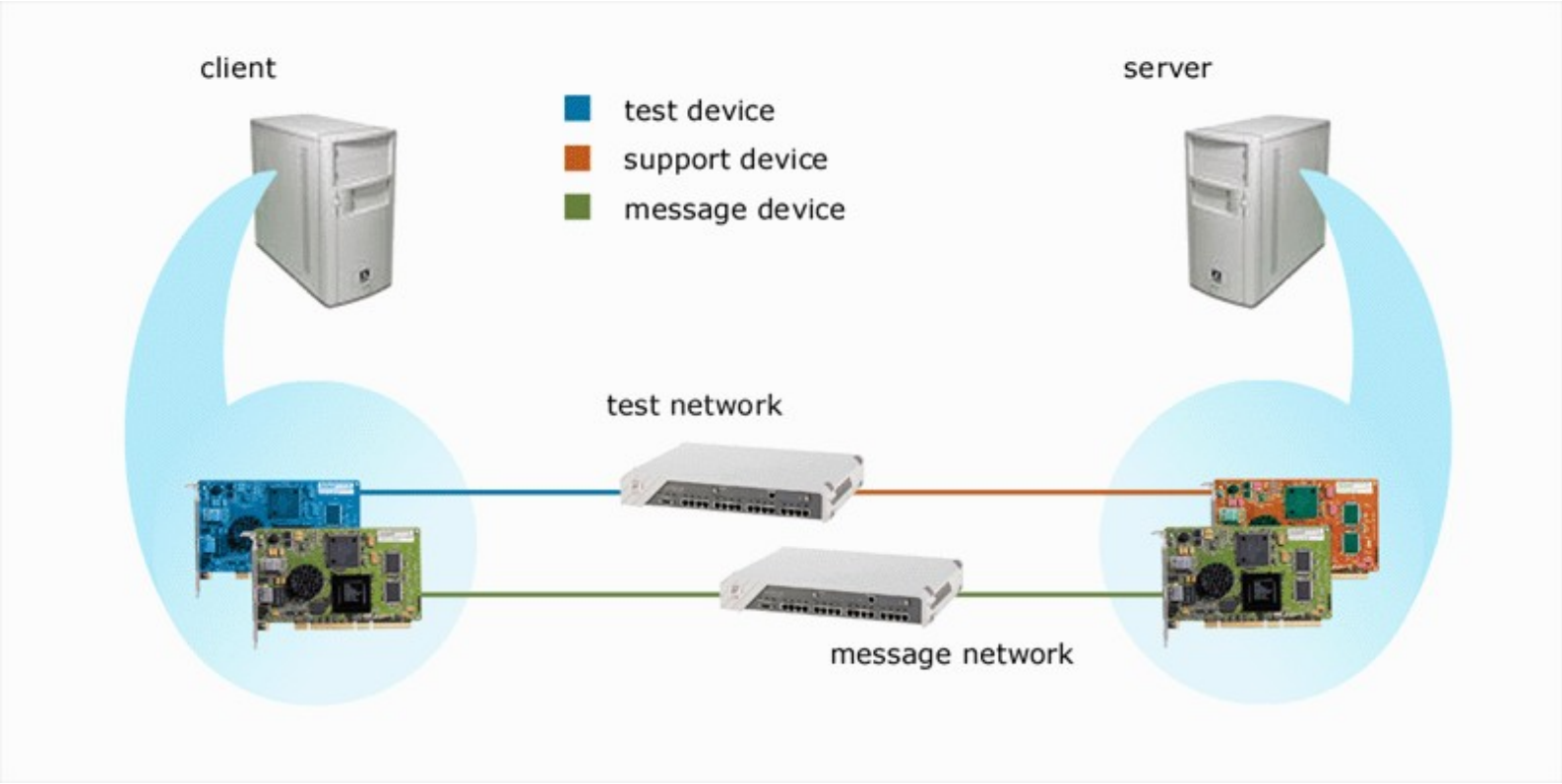
8.1 Architecture réseau Windows « user land »



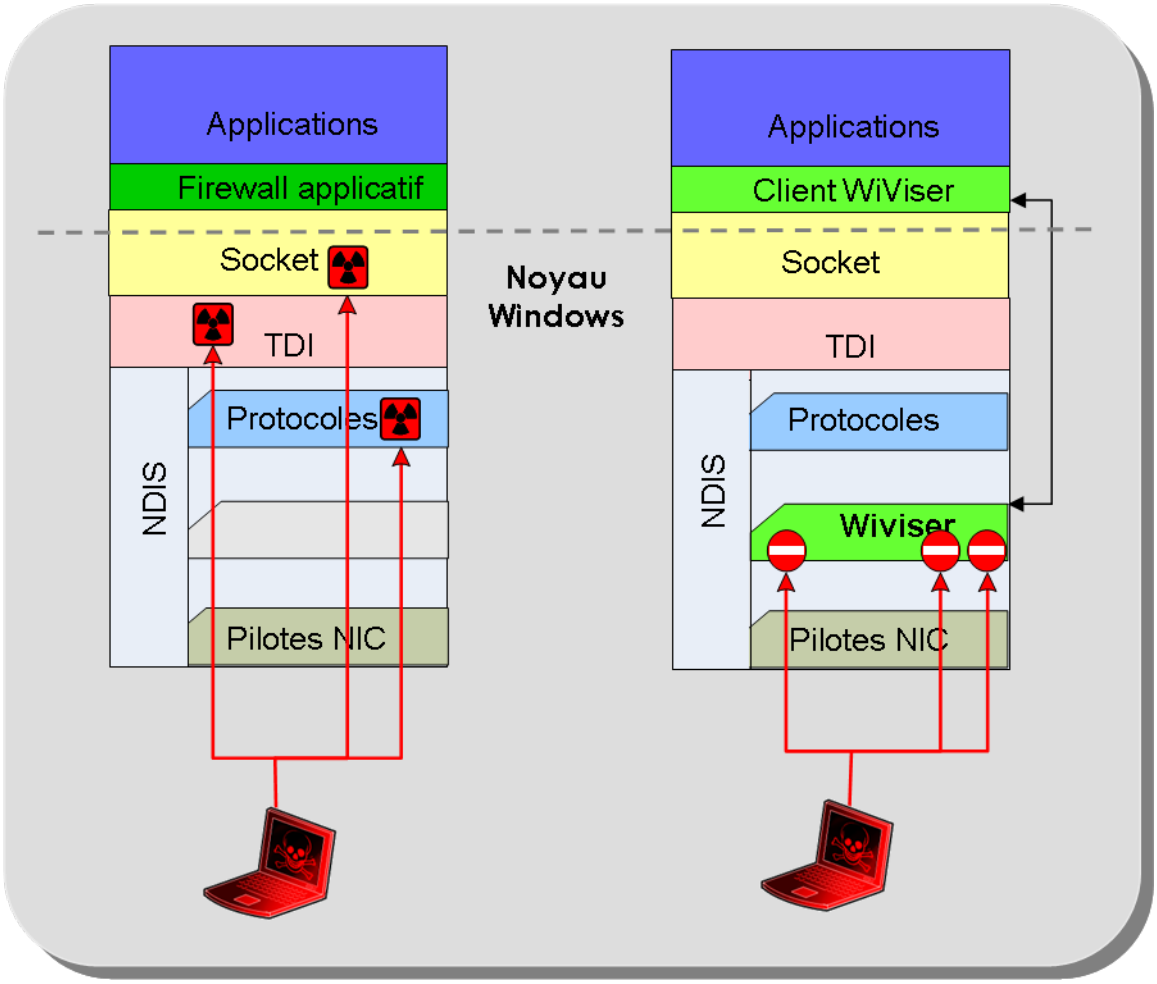
8.2 Architecture réseau Windows « kernel land »



8.3 Topologie des jeux d'essai pour le firewall



8.4 Protection du firewall réseau



8.5 Documentation syntaxique des règles de filtrage

Coté utilisateur, les règles de filtrage utilise le format XML.
(CF Annexe 8.6)

Chaque règle est contenue dans une balise rule. Cette balise contient deux paramètres:

Le paramètre action à deux valeur possible : accept et drop.
La valeur par défaut est accept.

action = "drop" : tout paquet vérifiant la règle est accepté.
action = "accept" : tout paquet vérifiant la règle est détruit.

Le paramètre status : deux valeur possible : true et false.
La valeur par défaut est false.

status = "true" : la règle est active.
status = "false" : la règle n'est pas interprétée par thor.

Chaque règle peut contenir des balises de filtrage pour spécifier le champ d'action.
Dans ces balises de filtrages sont définis des intervalles de filtrage définis par deux paramètres.
Une borne minimum dont le nom se termine par un 1 et une borne maximum dont le nom se termine par un 2.

La valeur par défaut d'une borne minimum est toujours la valeur minimum correspondante.
De même pour la valeur maximum.

Si l'intervalle est omit : la règles correspondra donc à la totalités des valeurs définis par l'ensemble.

Si un seul paramètre de l'intervalle est renseigné (min ou max) la règle agira que sur cet élément de l'ensemble.

- La balise de filtrage device permet de filtrer en fonction des propriétés physique de la communication.

Cette balise possède deux intervalles direction et adapter.

L'intervalle direction à deux valeurs possible input et output : direction1 = "input" : la règle concerne seulement les paquets entrant.

direction1 = "output" : la règle concerne seulement les paquets sortant.

L'intervalle adapter permet de spécifier l'identifiant des cartes Ethernet concernés par la règle :

adapter1 = "0" adapter2 = "5" : La règle concerne seulement les 6 premières cartes réseau.

- La balise de filtrage ethernet permet de filtrer les paquets au niveau de la couche transport.

Elle possède 3 intervalles :

L'intervalle srcaddr spécifie les adresses mac sources concernant la règle.

La syntaxe de l'adresse doit être conforme à ce format: HH:HH:HH:HH:HH:HH avec H étant un caractère hexadécimal en majuscule ou minuscule.

L'intervalle dstaddr spécifie les adresses mac de destination concernant la règle. La syntaxe de l'adresse est la même que pour l'intervalle srcaddr.

L'intervalle protocol permet de spécifier le champ des identifiants de protocoles réseaux (compris entre 0 et 65535) concernant la règle.

Si une balise de filtrage de protocoles réseau est spécifiée. L'optimisateur de règles se chargera de renseigner ce champ.

- La balise de filtrage ip permet de filtrer les paquets au niveau de la couche réseau.

Elle possède 3 intervalles :

L'intervalle srcaddr spécifie les adresses IP sources concernant la règle.

La syntaxe de l'adresse doit être conforme à ce format: u.u.u.u avec u étant un nombre compris entre 0 et 255.

L'intervalle dstaddr spécifie les adresses IP de destination concernant la règle. La syntaxe de l'adresse est la même que pour l'intervalle srcaddr.

L'intervalle protocol permet de spécifier le champ des identifiants de protocoles de transport (compris entre 0 et 255) concernant la règle.

Si une balise de filtrage de protocoles de transport est spécifiée. L'optimisateur de règles se chargera de renseigner ce champ.

Les balises de filtrage de transport sont tcp ip et icmp, leur présence dans une règle doit être exclusive et unique. Si ce n'est pas le cas, seul la première balise de filtrage est interprétée.

- La balise de filtrage tcp permet de filtrer les paquets respectant le protocole de transport du même nom. Elle contient deux intervalles : srcport et dstport.

L'intervalle srcport doit contenir l'intervalle de port source concernant la règle. (Compris entre 0 et 65535)

L'intervalle dstport doit contenir l'intervalle de port de destination concernant la règle. (Compris entre 0 et 65535)

- La balise de filtrage udp permet de filtrer les paquets respectant le protocole de transport du même nom. Elle contient deux intervalles : srcport et dstport.

L'intervalle srcport doit contenir l'intervalle de port source concernant la règle. (Compris entre 0 et 65535)

L'intervalle dstport doit contenir l'intervalle de port de destination concernant la règle. (Compris entre 0 et 65535)

- La balise de filtrage icmp permet de filtrer les paquets respectant le protocole de transport du même nom. Elle contient deux intervalles : type et code.

L'intervalle type doit contenir l'intervalle de type de requête ICMP concernant la règle. (Compris entre 0 et 255)

L'intervalle code doit contenir l'intervalle de code ICMP concernant la règle. (Compris entre 0 et 255)

L'ordre des règles de filtrage à une importance car le firewall analyse chaque règle de haut en bas, et s'arrête sur la première règle correspondant aux critères des paquets. (CF Annexe figure 1)

L'optimisateur de règle se charge de supprimer les règles qui sont masqué par des règles de plus grande priorité (c'est-à-dire des règles la précèdent ayant un intervalle de filtrage supérieur ou égale)

8.6 Exemple de regles de filtrage

```
<rule action = "drop" status = "true">
    <device direction1 = "input"/>
    <ip srcaddr1 = "192.168.1.102" protocol1="1"/>
</rule>

<rule action = "drop" status = "true">
    <device direction1 = "output"/>
    <ip dstaddr1 = "192.168.1.102" protocol1="1"/>
</rule>


<rule action = "drop" status = "false">
    <device
        direction1 = "input"
        adapter1 = "1"
        adapter2 = "4"
    />
    <ethernet
        srcaddr1 = "00:00:00:00:00:00"
        srcaddr2 = "33:33:33:33:33:33"
        dstaddr1 = "00:00:00:00:00:00"
        dstaddr2 = "33:33:33:33:33:33"
        protocol1 = "2048"
        protocol2 = "2048"
    />
    <ip
        srcaddr1 = "192.168.1.1"
        srcaddr2 = "192.168.1.1"
        dstaddr1 = "192.168.1.4"
        dstaddr2 = "192.168.1.4"
        protocol1 = "174"
        protocol2 = "17"
    />
    <udp
        srcport1 = "666"
        srcport2 = "666"
        dstport1 = "0"
        dstport2 = "65535"
    />
</rule>
```

```
<rule action = "drop" status = "false">
    <device
        direction1 = "output"

    />
    <icmp
        type1 = "4"
        type2 = "4"
        code1 = "0"
        code2 = "10"
    />
</rule>

<rule action = "drop" status = "false">
    <device
        direction1 = "output"
adapter1 = "1"
    />
    <icmp
        type1 = "4"
        type2 = "4"
        code1 = "0"
        code2 = "1"
    />
</rule>


<rule action = "accept" status = "false">
    <tcp
        srcport1 = "3098"
        srcport2 = "3098"
        dstport1 = "0"
        dstport2 = "65535"
    />
</rule>

<rule action = "drop" status = "true"/>

<rule action = "drop" status = "true">
    <device direction1 = "input"/>
    <ip srcaddr1 = "192.168.1.1"/>
</rule>
```

8.7 Algorithme d'un système de filtrage