Pascal MIETLICKI

Licence Professionnelle Techni.com
Université de la Polynésie Française

Mise en place d'une architecture de contrôle de la sécurité

Cellule des Ressources Informatiques :: Université de la Polynésie Française

Octobre 2005

- Pascal MIETLICKI
- Année 2004/2005
- Cellule des Ressources Informatiques de l'Université de Polynésie Française
- Maître de stage et enseignant-tuteur : Franck MEVEL

- Sujet du stage
  Mise en place d'une architecture de contrôle de la sécurité du réseau de l'Université de Polynésie Française

---

- 1 PC sous linux debian Sid pour exploitation
- 2 PC sous linux debian Sarge pour mise en production

Table des matières

Remerciements

- Mon maître de stage et enseignant-tuteur : **Franck MEVEL** qui m'a fait bénéficier de son expérience et m'a appuyé tout au long des étapes de ce stage.

- Mes collègues de la CRI pour leur solidarité

Introduction générale

Les menaces qui pèsent sur les ressources d'informations prennent différentes formes. La sécurité des informations peut être compromise par des moyens très simples.
Les menaces sur le réseau appartiennent à deux catégories : interne et externe. Dans la plupart des organisations, la sécurité du réseau peut être comparée à un oeuf : la coque extérieure est difficile à pénétrer mais, une fois cette limite franchie, l'intérieur est tendre et n'offre aucune résistance. En majorité les réseaux sont protégés par des pare-feu au monde extérieur, ce qui constitue une véritable forteresse pour tout pirate. Les choses sont très différentes à l'intérieur. Les communications confidentielles non cryptées, les hôtes incorrectement maintenus et l'absence de contrôle de sécurité logique facilitent le déroulement d'une attaque et la rende même plus difficile à détecter. Les statistiques montrent que 80% des attaques fructueuses sont internes.

Dans le domaine de la détection d'intrusion, il est important d'identifier les origines probables des attaques. En connaissant ces origines, on peut déployer la détection d'intrusion aux emplacements les plus efficaces.

En réponse à cette problématique, il est nécessaire d'instaurer une architecture de contrôle de la sécurité afin de déceler les attaques et les failles potentielles internes pour y remédier mais également détecter les attaques externes pouvant aboutir à une intrusion interne. Ceci constitue le but de ce stage.

Ce rapport couvre la théorie et la pratique des IDS (Intrusion Detection System) et des outils de sécurité en général avec Prelude-IDS comme fil conducteur.
Il présente le travail effectué pour la mise en place de l'ensemble des outils logiciels nécessaires à la sécurité du réseau :
Nessus pour la découverte des vulnérabilités existantes et leurs résolutions
Prelude-manager pour la centralisation des alertes
Prelude-lml pour la détection des intrusions hôtes
Snort pour la détection des intrusions réseaux
Honeyd pour la génération de machines « leurres »
Samhain pour veiller à l'intégrité des systèmes

I. Objectif du stage

L'objectif de ce stage est de sécuriser le réseau interne de l'Université. Le système à mettre en place sera chargé d'identifier les failles, de les résorber mais aussi de prévenir les attaques.
Avant de présenter l'étude dans son ensemble, nous allons voir ce qu'est la sécurité en général. Nous pouvons découper le domaine de la sécurité de la façon suivante :

- La sécurité logicielle qui gère la sécurité du Système d'Information au niveau logiciel et qui intègre des protections comme les antivirus.
- La sécurité du personnel qui comprend la formation et la sensibilisation des personnes utilisant ou travaillant avec le Système d'Information.
- La sécurité physique qui regroupe la politique d'accès aux bâtiments, la politique d'accès au matériel informatique et les règles de sécurité pour la protection des équipements réseaux actifs et passifs.
- La sécurité procédurale définit les procédures et les règles d'utilisation du Système d'Information.
- La sécurité réseau où sont gérés l'architecture physique et logique du réseau, les politiques d'accès aux différents services, la gestion des flux d'informations sur les réseaux et surtout les points de contrôles et de surveillance du réseau.
- La veille technologique qui est souvent oubliée et qui permet de faire évoluer la sécurité au cours du temps afin de maintenir un niveau de protection du système d'information suffisant.

Après avoir vu le découpage organisationnel de la sécurité, intéressons-nous aux objectifs. Une bonne politique de sécurité doit préserver les aspects de :

- Disponibilité : c'est-à-dire fournir l'accès à l'information pour que les utilisateurs autorisés puissent la lire ou la modifier. Ou encore faire en sorte qu'aucune personne ne puisse empêcher les utilisateurs autorisés d'accéder à l'information.
- Confidentialité : c'est-à-dire empêcher les utilisateurs de lire une information confidentielle (sauf s'ils y sont autorisés). Ou encore, empêcher les utilisateurs autorisés à lire une information, de la divulguer à d'autres utilisateurs (sauf autorisation).
- Intégrité : c'est-à-dire empêcher une modification (création, mise à jour, ou destruction) indue de l'information ou encore faire en sorte qu'aucun utilisateur ne puisse empêcher la modification légitime de l'information.

Toute entrave à la sécurité peut être modélisée de la façon suivante :

CAUSE	*ETAT*	*SERVICE*
Faute →	Erreur →	Défaillance

Faute : Cause adjugée ou supposée d'une erreur.
Erreur : Au moins une partie du système suit un comportement erroné, susceptible d'entraîner une défaillance.
Défaillance : Le service délivré par le système dévie du service spécifié. L'incident de sécurité est arrivé à son terme.

Afin de garantir une sécurité suffisante, il faut que toute attaque soit bloquée pendant l'une des 3 phases (et surtout avant la fin de la troisième). Généralement, plus on se rapproche de la défaillance, plus le problème sera difficile et long à résoudre... et l'on se rapproche de la réussite de l'attaque.

A) Menaces internes

Les attaques internes représentent la majorité des attaques fructueuses dans une infrastructure réseau. Elles sont parfois destructrices et souvent plus difficile à déceler. Les initiés de l'organisation, qui connaissent parfaitement le fonctionnement des contrôles de sécurité et qui disposent de tout le temps nécessaire pour planifier l'attaque, représentent un facteur aggravant de la situation. Ces initiés peuvent utiliser l'accès légitime dont ils disposent déjà pour obtenir un accès supplémentaire non autorisé aux systèmes.

Les attaques internes sont plus difficiles à détecter que les attaques externes. Elles se produisent lorsque les organisations ne surveillent pas l'intérieur de leur infrastructure aussi soigneusement que l'extérieur. Une attaque interne peut être menée par un employé qui aura accumulé graduellement des accès et informations privilégiés pendant plusieurs années, voire plusieurs dizaines d'années. Ce type d'agissements semble démesuré mais il est particulièrement vrai pour les grandes organisations qui suscitent un certain engouement de la part de leurs concurrents. Mettre en place un tel système « garde-fou » de sécurité peut sembler exagérer pour une institution tel qu'une université mais l'on est jamais à l'abri d'un individu malveillant ou imprudent qui aurait malencontreusement déployé du code viral au sein du réseau.
D'autre part, les utilisateurs peuvent compromettre la sécurité interne en installant des applications de messagerie instantanée et de partage de fichiers P2P (point à point) qui mettent en défaut le pare-feu. Certaines applications P2P sont distribuées avec des logiciels espions ou des fonctions qui activent en secret le partage de la totalité du disque dur. Des utilitaires de messagerie instantanée orientés proxy, comme celui d'AOL, peuvent être utilisés pour s'infiltrer dans n'importe quel port ouvert d'un pare-feu interne. Les virus modernes sont transmis par de nombreuses attaques capables d'ouvrir un système pour les y introduire. La plupart des utilisateurs standard n'auront pas conscience d'ouvrir une brèche dans la sécurité en effectuant leurs activités quotidiennes.

B) Comment intervenir pour une meilleure sécurité du réseau interne?

1. Découvrir les failles existantes

Il convient de faire un « état des lieux » du réseau. En effet, à l'heure actuelle un ensemble d'attaques et de failles de sécurité sont connues. Des rapports sont publiés tous les jours concernant de nouvelles failles. Il est donc nécessaire de disposer d'un système permettant d'analyser l'ensemble du réseau afin de mettre en évidence ces mêmes failles et d'indiquer le niveau où elles interviennent. Ce système doit aussi fournir des outils de mise à jour afin de laisser un temps minimum entre la découverte de la faille et son exploitation par d'éventuels hackers. Le stage, une fois abouti, devra répondre à cette problématique.

2. La solution des IDS

Un IDS (Intrusion Detection System) peut être utilisé en interne comme en externe pour détecter à la fois les attaques orchestrées de l'extérieur et les attaques internes délibérées. Il est capable de détecter la signature de la plupart des outils P2P, les utilisations inappropriées d'Internet et les messageries instantanées. Ces fonctionnalités complètent la surveillance de base attendue. Ces caractéristiques font de l'IDS interne une application extrêmement puissante.

3. Vérification d'intégrité

La vérification d'intégrité est une méthode simple et particulièrement efficace pour surveiller les intrusions. Elle repose sur la génération d'un total de contrôle pour chaque fichier d'un système, puis sur la comparaison périodique de ce total avec celui du fichier original afin de vérifier qu'aucune modification n'a été apportée. Dès que l'on détecte un changement non autorisé le système génère une alerte. Quel que soit le système, de nombreux fichiers changent régulièrement dans le cadre de l'activité normale. Ainsi l'utilitaire de vérification d'intégrité doit être soigneusement paramétré afin d'éviter les fausses alertes. Les totaux de contrôle doivent être redéfinis dès que des changements légitimes se produisent.
La vérification d'intégrité détecte les dégradations de page Web. Les attaquants obtiennent souvent l'accès à des serveurs Web externes en liaison avec le réseau et changent le contenu qu'ils affichent. Un vérificateur d'intégrité peut être déployé entre autre pour surveiller des fichiers de page Web spécifiques. Lorsque l'attaquant change le contenu d'une page Web, la vérification du total de contrôle échoue et le correspondant approprié est averti. Les fichiers d'un tel serveur Web ne changent pas assez souvent pour risquer de générer un déluge de faux positifs. On peut aussi configurer le vérificateur pour rétablir automatiquement les fichiers dans leur état initial.

La vérification d'intégrité possède bien sûr aussi ses limites. Le principal inconvénient de cette technologie est qu'elle nécessite d'accéder à des fichiers essentiels sur l'hôte surveillé. Cela peut être problématique en cas de failles au sein du vérificateur d'intégrité. Il est aussi possible de modifier les totaux de contrôle pour correspondre au fichier originel convoité ce qui annule l'action par vérification d'intégrité. On peut toutefois réduire ce risque en stockant les totaux de contrôle sur un serveur dédié bien protégé, mais il est toujours possible qu'un hacker corrompe le serveur, la sécurité absolue est une utopie.

4. La solution de « pot de miel »

Une méthode complémentaire consiste à attirer les pirates potentiels sur de fausses cibles, d'où la métaphore du pot de miel où les pirates sont comparés à des abeilles appâtées sur des cibles idéales (le pot de miel) mais qui, finalement, s'enlisent dans le piège.
Le pot de miel est un concept intéressant qui permet de créer des serveurs virtuels qui contiennent des vulnérabilités intentionnelles. Ainsi, si un pirate parvient à s'introduire sur le réseau, il sera inexorablement attiré par les machines les plus fragiles. Hors, avec le pot de miel, ces machines sont délibérément vulnérables et, dès que le pirate s'attaque à ses machines virtuelles, une alerte est envoyée aux administrateurs réseaux afin qu'ils prennent les dispositions nécessaires.
D'un point de vue pédagogique, on peut exploiter ce système afin d'étudier les agissements du pirate qui a réussi à infiltrer le réseau et mieux déterminer son profil.

Toute ces solutions constituent des moyens efficaces pour protéger son réseau des attaquants éventuels et constituent la problématique de ce stage : **la mise en place d'une architecture de sécurité du réseau** comprenant l'ensemble de ces concepts. Ils permettent de positionner la sécurité à un niveau plutôt élevé mais ils ne constituent pas une solution miracle. La principale difficulté réside dans le fait qu'il est nécessaire d'effectuer une étude préalable afin de savoir l'endroit où ces systèmes seront les plus efficaces. D'autre part, ils demandent un paramétrage très précis afin de ne laisser s'échapper aucune alerte importante. La sécurité absolue n'existe pas.

5. Choix des outils

Les outils mis en place ont été sélectionnés sur plusieurs critères. Tout d'abord, pour leur efficacité par rapport aux autres outils du marché. Ensuite, pour leur caractère libre avec tous les avantages liés tel que l'adaptabilité, l'évolutivité, etc. De plus, ils sont gratuits ce qui est un avantage non négligeable. Le dernier critère de sélection est leur aptitude à contrôler des réseaux hétérogènes (multi-plateformes).

Mise en oeuvre du stage

Le réseau de l'université ne fait pas exception à la règle, il est bien protégé de l'extérieur mais dispose de peu de mécanismes de sécurité au sein des réseaux délimités.

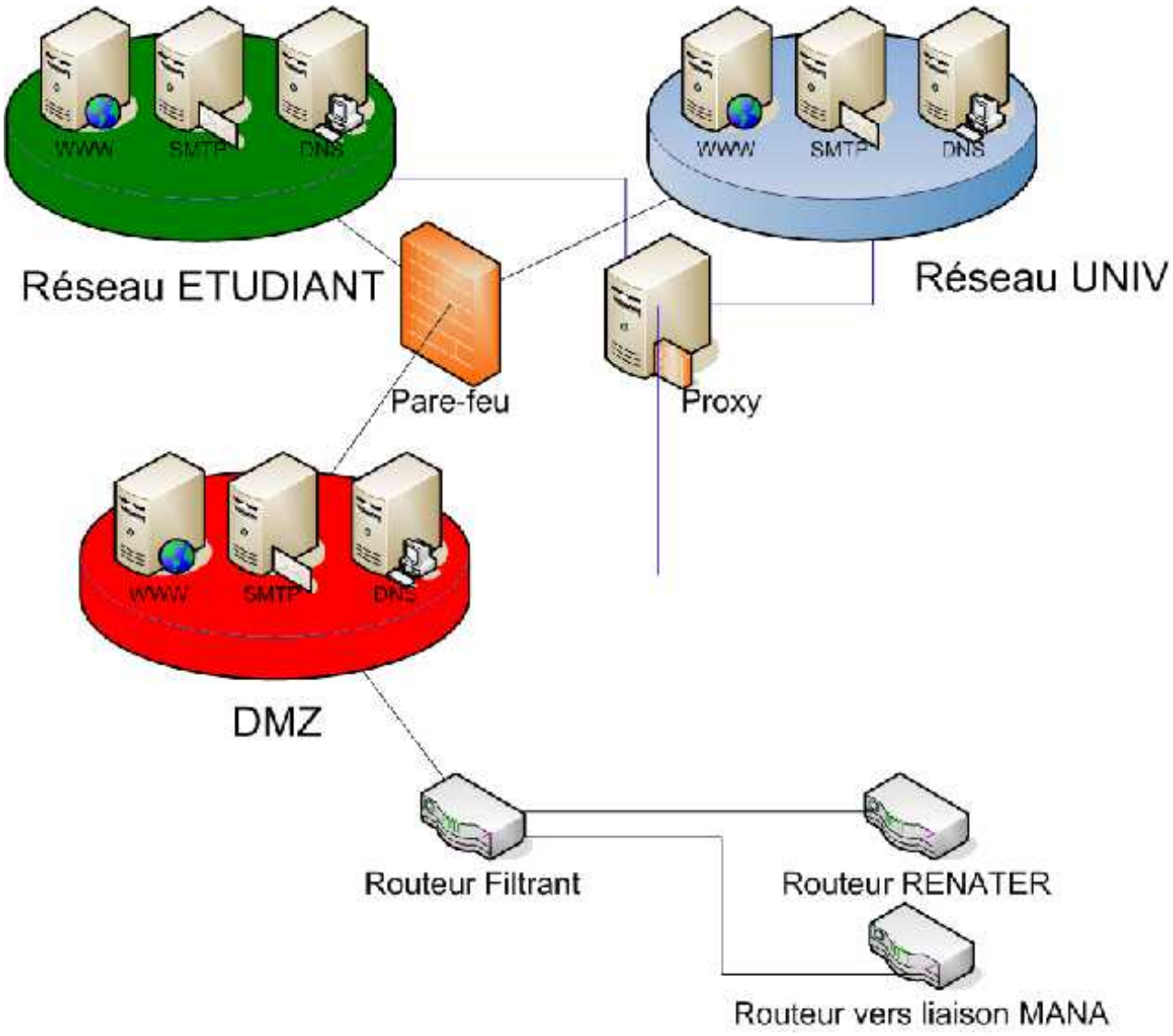


Figure 1: Réseau de l'Université de la Polynésie Française

Sur ce schéma, on remarque que le réseau interne est protégé à la fois par un routeur filtrant et un pare-feu. Chaque zone est bien délimitée mais, à l'intérieur de ces zones, peu de mécanismes permettant d'assurer la sécurité et de prévenir les attaques existent.

Dans un premier temps, nous allons mettre en place les outils nous permettant de sonder l'état du réseau interne puis déployer les correctifs nécessaires.

Ensuite, nous nous intéresserons à la prévention des intrusions à la fois internes et

externes.

I. Les scanners de vulnérabilités

A) Généralités

Un scanner de vulnérabilité est un utilitaire permettant de réaliser un audit de sécurité d'un réseau en effectuant un balayage des ports ouverts sur une machine donnée ou sur un réseau tout entier. Le balayage se fait grâce à des sondes (requêtes) permettant de déterminer les services fonctionnant sur un hôte distant.
Il est possible avec ce type d'outil de lancer une analyse sur une plage ou une liste d'adresses IP afin de cartographier entièrement un réseau.

1. Fonctionnement d'un scanner

Un scanner de vulnérabilité est capable de déterminer les ports ouverts sur un système en envoyant des requêtes successives sur les différents ports et analyse les réponses afin de déterminer lesquels sont actifs.
En analysant très finement la structure des paquets TCP/IP reçus, les scanners de sécurité évolués sont parfois capables de déterminer le système d'exploitation de la machine distante ainsi que les versions des applications associées aux ports et, le cas échéant, de conseiller les mises à jour nécessaires, on parle ainsi de caractérisation de version.
On distingue habituellement deux méthodes :
L'acquisition active d'informations consistant à envoyer un grand nombre de paquets possédant des en-têtes caractéristiques et la plupart du temps non conformes aux recommandations et à analyser les réponses afin de déterminer la version de l'application utilisée. En effet, chaque application implémente les protocoles d'une façon légèrement différente, ce qui permet de les distinguer.
L'acquisition passive d'informations (appelé balayage passif) est beaucoup moins intrusive et risque donc moins d'être détecté par un système de détection d'intrusions. Son principe de fonctionnement est proche, si ce n'est qu'il consiste à analyser les champs des datagrammes IP circulant sur un réseau, à l'aide d'un sniffer. La caractérisation de version passive analyse l'évolution des valeurs des champs sur des séries de fragments, ce qui implique un temps d'analyse beaucoup plus long. Ce type d'analyse est ainsi très difficile voire impossible à détecter.

2. Utilité d'un scanner

Les scanners de sécurité sont des outils très utiles pour les administrateurs système et réseau afin de surveiller la sécurité du parc informatique dont ils ont la charge.

B) Identifier les vulnérabilités : Nessus

L'une des méthodes les plus courantes de pénétration consiste à utiliser les vulnérabilités connues des systèmes d'exploitation et des réseaux. Faiblesses auxquelles on peut généralement remédier par des patchs ou des modifications mineures de la configuration. Il est donc important d'implémenter, à temps, des procédures permettant de découvrir, évaluer et atténuer les vulnérabilités de sécurité grâce à des outils tel que Nessus, le scanner d'évaluation libre.
Nessus a été écrit par Renaud Deraison, auteur open source vivant à Paris en 1998. Cet outil, porté par le mouvement open source, a rapidement égalé voir surpassé ses concurrents commerciaux à tel point que bon nombre de professionnels « enrobent » du Nessus pour mieux vendre leur service.
Nessus emploie un modèle de plug-in extensible qui permet à la communauté de la sécurité d'ajouter à la demande des modules d'exploration.
Nessus se compose de deux applications. Premièrement, le serveur, qui se charge de toutes les attaques sur les machines auditées. Deuxièmement, le client, qui se charge de fournir une interface de dialogue entre l'homme et le serveur. Le serveur ne fonctionne que sur Unix (il existe des versions pour les systèmes de Microsoft mais instables actuellement), le client cependant peut être utilisé sur Unix et sur Windows. La transmission est sécurisée grâce à l'établissement préalable d'un certificat de sécurité ainsi que d'une connexion ssh.
Les sorties générées par Nessus, appelées rapports, peuvent servir de base à un audit de serveur par exemple. Plusieurs formats de sortie sont disponibles (pdf, html, texte) afin de satisfaire au mieux les besoins.

1. Démarrage du serveur

L'installation ainsi que les paramètres de configuration du serveur et du client sont disponibles en Annexe.
Afin d'automatiser le démarrage du serveur, j'ai développé un script de mise en route automatique au démarrage de la machine.

2. Comment fonctionne Nessus

a) Séquence des opérations

Nessus détermine les ports ouverts :
- en appelant le scanner externe nmap,
- en appelant snmpwalk,

(Le fait qu'une machine divulgue des informations par netstat ou SNMP est une faiblesse. Toutefois, on peut configurer ces services pour ne répondre qu'à un petit nombre de machines de supervisions, dont le scanner Nessus.)
- en se connectant au service netstat ou en exécutant netstat sur la machine s'il

dispose d'un accès SSH,
- en utilisant un plugin interne, calqué sur un des modes de fonctionnement de nmap, ou en utilisant un fichier externe, vu comme un résultat de nmap, obtenu par un moyen quelconque, par exemple en convertissant la sortie de la commande netstat.

Ensuite le plugin "find_service" tente d'identifier les processus connectés sur chaque port :
- tout d'abord en tentant des connexions SSL (TLSv1, SSLv3, SSLv2) puis standard
- ensuite en envoyant diverses séquences au service et en regardant les réponses.

find_service stocke ses découvertes sous forme de "clés" dans la "base de connaissance" (KB).
Sur chacun des ports ouverts, Nessus tente alors diverses attaques. Par exemple, si un script vise les serveurs Web, il sera lancé contre tous les ports où tourne un serveur HTTP ou HTTPS.

b) Types et fonctionnements des plugins

Les plugins de Nessus sont classés par famille. Ceci permet de les regrouper dans l'interface graphique mais n'a aucune influence sur le fonctionnement du scanner. Il existe une famille "déni de service". Certains scripts de cette famille ne sont pas dangereux : ils vérifient la présence d'un logiciel vulnérable à un déni de service sans le tuer. Chaque plugin fait partie d'une "catégorie" :
- ACT_INIT

Ces script servent simplement à configurer des options et ne fait aucun test.
- ACT_SCANNER

Ces scripts servent un scanner de ports ou apparenté (e.g. ping).
- ACT_SETTINGS

Même fonction qu'ACT_INIT, mais passe après les scanners, quand on est sûr que la machine répond.
- ACT_GATHER_INFO

Ces script récupèrent des informations sur le système, par exemple identifient les services ou vérifient la présence d'un logiciel particulier.
- ACT_ATTACK

Ce scripts tentent de percer certaines défenses, en théorie sans effet pervers contre la disponibilité du système.
- ACT_MIXED_ATTACK

Ces script constituent un « intermédiaire » entre une attaque de sévérité basse et hautes. Ils sont susceptibles d'avoir des effets secondaires désastreux, bien que ce ne soit leurs buts.
- ACT_DESTRUCTIVE_ATTACK

Ces scripts tentent de perturber un service ou détruire des données.
- ACT_DENIAL

Ces scripts tentent un déni de service contre un logiciel particulier.
- ACT_KILL_HOST
Ces script tentent un déni de service contre la machine/le système d'exploitation.
- ACT_FLOOD
Ces scripts tentent un déni de service par envoi massif de paquets et peuvent perturber l'ensemble du réseau.
- ACT_END
Ces scripts se contentent de compiler les informations une fois le scan passé.
La frontière entre toutes ces catégories est floue, et il est impossible de prédire a priori si un script qui vise un logiciel donné n'aura pas des effets dangereux contre un autre.
Nessus exécute en premier les scripts ACT_INIT, puis les plugins ACT_SCANNER, ACT_SETTINGS, ACT_GATHER_INFO, etc.
Enfin, chaque script déclare des "dépendances"
- en terme de scripts qui ont dû tourner auparavant.
(Par exemple, la plupart des scripts dépendent de "find_service")
- en terme de ports/services ouverts
Par exemple, les scripts qui testent des vulnérabilités HTTP déclareront dépendre du port 80 et de la clé "Services/www".
Par principe, Nessus ne considère rien comme acquis. Contrairement à certains scanners de sécurité qui basent leurs vérifications sur les bannières présentées, Nessus attaque réellement les services, sauf si l'option "safe checks" est active.
En conséquence :
- Nessus est capable de détecter si une faille, censée être corrigée dans la version N+1 d'un logiciel, est toujours là.
- Nessus peut découvrir qu'une faille dirigée contre le produit X fonctionne aussi contre le produit Y.

c) Comment un test de sécurité peut tuer votre système

i-Les dangers du scan de ports

En TCP, le scanner ouvre une connexion puis la referme immédiatement sans envoyer de données. Certains logiciels meurent ou partent en boucle s'ils n'arrivent pas à lire de données.
En UDP, il envoie un paquet sans données. Ceci est suffisant pour tuer certaines piles IP défectueuses ou un logiciel mal codé.

ii-Entreprise de démolition Nessus & Cie

Certains scripts génériques sont particulièrement méchants :
- Débordements mémoire contre divers champs/requêtes des protocoles HTTP, FTP, POP3...

- Requêtes mal formées (HTTP, FTP...)
- Tests par saturation, inondant un service inconnu de myriades d'octets. Par ailleurs, outre les effets secondaires du scanner de port, certains logiciels n'apprécient pas l'interrogatoire que leur fait subir find_service, à commencer par les multiples tentatives de connexion SSL.
Il existe un test qui pourrait réellement tenter d'effacer des données : http_methods.nasl. Toutefois, la partie de code dangereuse est désactivée en mode "safe checks".

iii-Limitation des risques

Activez l'option "safe checks"
Activez "optimize the test".
Désactivez "enable dependencies at run time".
Supprimez les plugins que vous pensez inutiles ou dangereux.
Si aucun service ne s'appuie sur SSL, désactivez "test SSL services"
désactivez check_ports.nasl

3. Bilan

Nessus est un outil de test, permettant d'indiquer aux administrateurs réseau les correctifs qu'ils doivent appliquer pour durcir leur système.
Il permet de tester avec un maximum d'efficacité les mesures de protection mises en place.
En bref, Nessus constitue un outil prépondérant afin de tester la fiabilité de son réseau. Il ne se contente pas seulement de tester les machines, il les attaque effectivement. Ce mécanisme qui simule les attaques réelles des pirates assure la pertinence des rapports générés.

4. Analyse du réseau

Nessus a analysé avec succès le réseau de l'Université et a permis de soulever certains problèmes. Les serveurs étaient correctement à jour mais les postes clients internes contenaient un certain nombre de failles de sécurité liées à des correctifs non installés. J'ai été chargé de trouver des solutions pour la mise à jour automatique des postes de travail de l'Université.

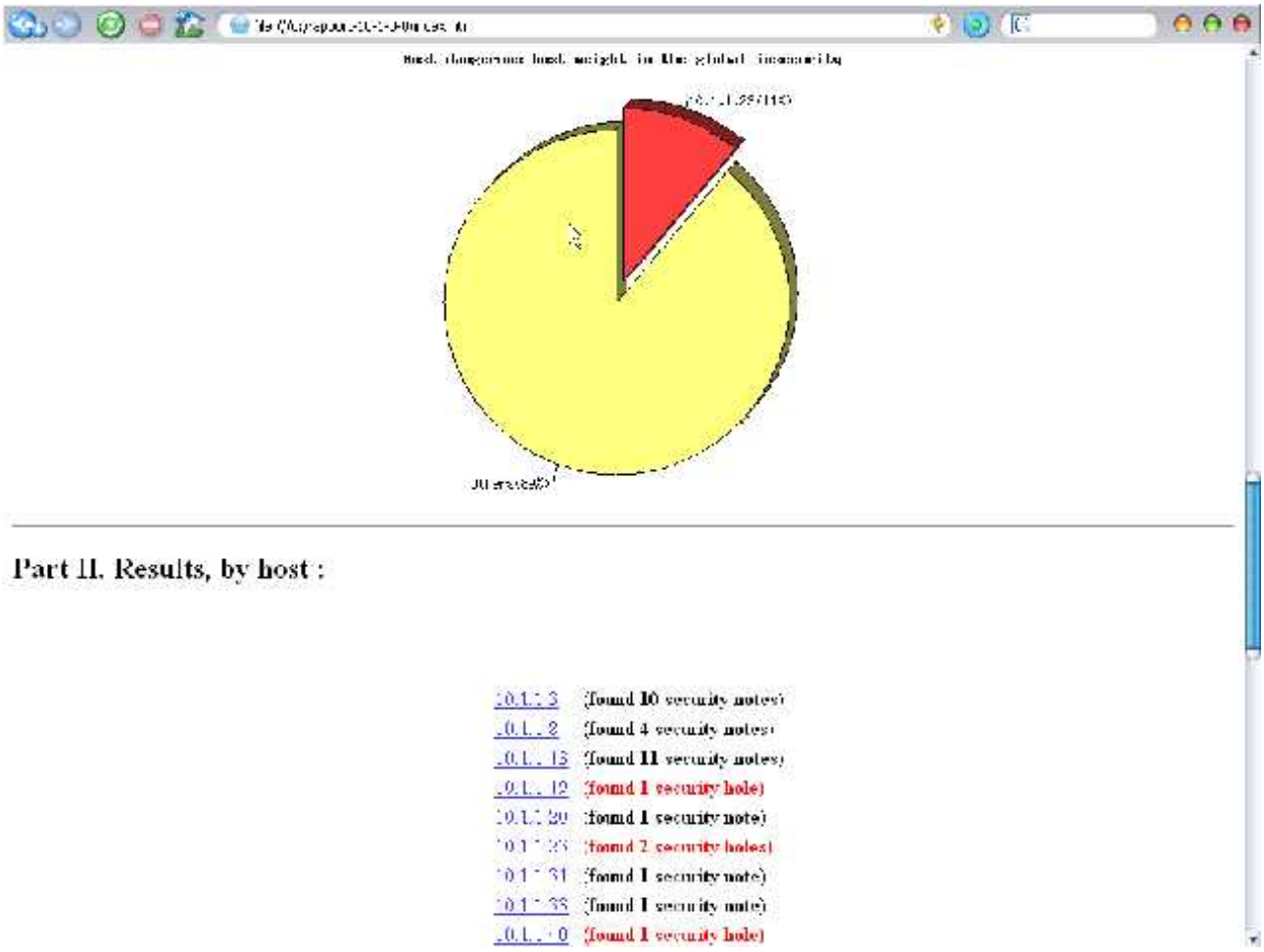


Figure 2: Rapport Nessus pour l'analyse du réseau de l'UPF

a) WSUS (Windows Server Update Services)

i-Objectifs

Les mises à jour sont normalement effectuées à partir d'un serveur de Microsoft donc par internet. Dans le cas d'un parc informatique conséquent, la mise à jour surchargerait inutilement la bande passante très faible de l'Université. Un serveur SUS (Software Update Services) est déjà en place sur le campus; il remplace le serveur de Microsoft. Les correctifs (validés par l'administrateur) sont centralisés sur cette machine. Les machines clientes ayant besoin d'une mise à jour les récupèrent alors directement du serveur. Ainsi le parc informatique dispose des derniers correctifs de sécurité. L'analyse précédente par Nessus a montré que les mises à jour ne sont pas déployées correctement. Il convient donc de mettre en place un outil plus fiable permettant le contrôle du déploiement.

ii-Améliorations par rapport à SUS

WSUS permet une administration complète et en temps réel du parc informatique. Il est multi-langages et les correctifs peuvent être paramétrés pour être installés ou

désinstallés de manière différée.
Il génère entre autres des rapports complets permettant de situer le déroulement du processus de mise à jour ainsi qu'une interface simple de gestion de ce processus.

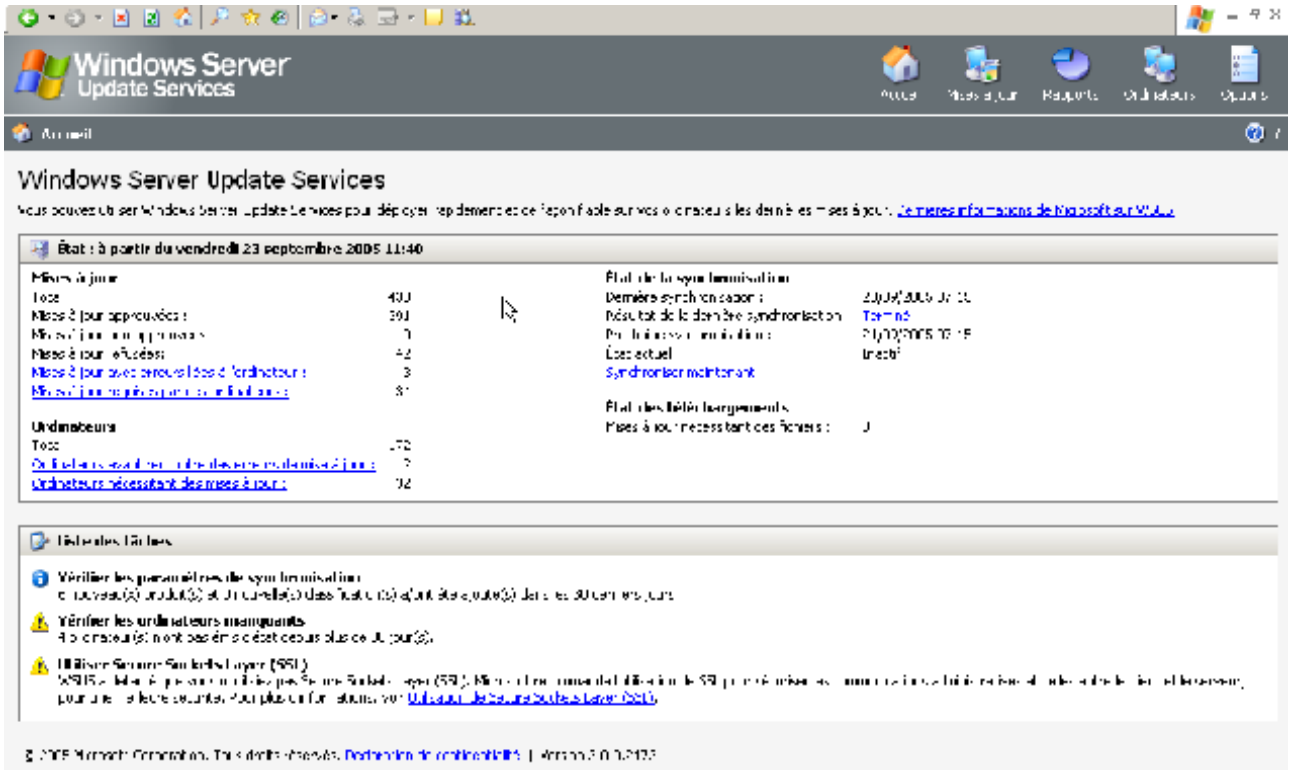


Figure 3: Interface Web WSUS de gestion des mises à jour

b) Apt-proxy pour Linux

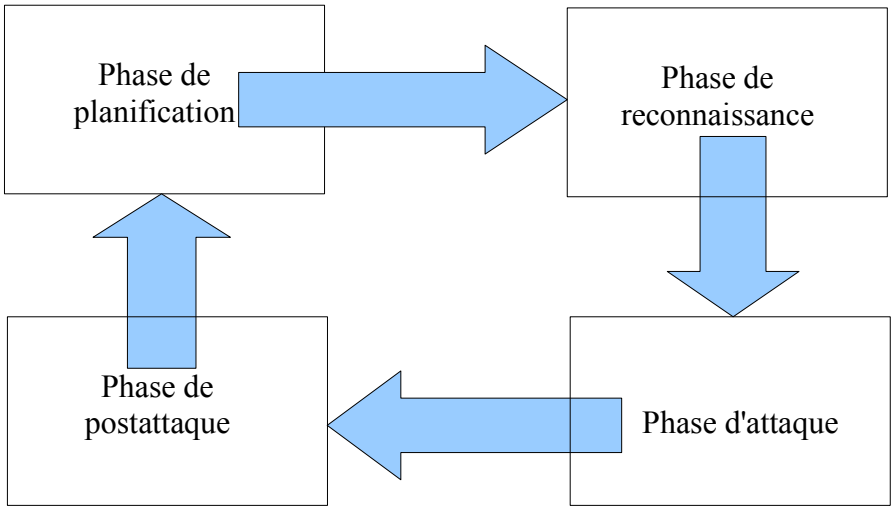
Nous avons installé ce serveur de mise à jour pour les ordinateurs fonctionnant sous Linux. Apt-proxy fonctionne sur le même principe que SUS : la centralisation des mises à jour sur un serveur unique. Apt-proxy est un logiciel serveur permettant de simuler un miroir debian. Pour les clients, seul un paramétrage permettant d'indiquer le serveur de mise à jour doit être fait. L'inconvénient de cette technique est la même que pour SUS, c'est à dire qu'il n'y a pas de contrôle du déploiement des correctifs. A l'avenir, il serait intéressant de développer une application permettant le contrôle des clients.

II. Prévention des attaques

Après avoir mis en place un système de détection des failles existantes du réseau et effectuer les mises à jour nécessaires en conséquence, il faut maintenant déployer un système de prévention des intrusions permettant de sécuriser le réseau des attaques internes ou externes. Tout d'abord, nous allons voir les différents mécanismes constituant une attaque.

A) Orchestration d'une attaque

Plusieurs phases de l'orchestration d'une attaque sont suffisamment génériques pour concerner la plupart d'entre elles (pour plus d'informations, les détails d'une attaque sont fournis en annexe).



1. Phase de planification

Les pirates planifient généralement à l'avance l'attaque d'un système. Cette planification peut prendre diverses formes. L'attaquant utilise souvent le système d'une façon normale pour l'étudier avant de lancer les hostilités. Ce type d'accès légitime ouvert au public lui permet d'établir la portée et les objectifs de l'attaque.

2. Phase de reconnaissance

L'attaquant récupère ensuite des informations ou effectue une reconnaissance de votre réseau. Il va émettre différentes requêtes dont l'objectif est de définir une méthode d'attaque spécifique.

3. Phase d'attaques

Après la phase initiale de planification et de reconnaissance, la phase suivante logique

consiste à exploiter les informations obtenues et à attaquer le réseau. Le trafic généré dans ce cas peut prendre diverses formes. Tout ce qui ressemble à du code d'exploitation distant en passant par le trafic normal suspect peut signaler une tentative d'attaque qui exige une action.
Il existe plusieurs types d'attaques possibles :

a) Déni de service

On classe dans cette catégorie les attaques visant à rendre indisponible pendant un temps indéterminé les services ou ressources d'une organisation soit en épuisant les ressources (saturation du service) soit en envoyant des paquets malveillants.

b) Exploitation distante

Ce type d'attaque exploite les failles des logiciels présents en profitant des entrées mal contrôlées ou des erreurs de configuration.

c) Chevaux de Troie(Trojan) et entrée secrète (Backdoor)

En les installant, un hacker peut outrepasser les contrôles de sécurité normaux et obtenir un accès privilégié à l'hôte. Afin de les déployer, il peut utiliser du code viral introduit, par exemple, dans la pièce jointe d'un courrier électronique.

4. Phase postattaque

Une fois qu'un attaquant a réussi à infiltrer un hôte sur le réseau, les actions qu'il va ensuite entreprendre ne suivent aucun modèle prévisible. C'est au cours de cette phase que l'attaquant va élaborer son plan et exploiter les ressources d'informations comme il le jugera bon. Voici quelques options qui s'offrent à lui à ce niveau des opérations:
dissimuler ses traces
pénétrer plus loin dans l'infrastructure réseau
se servir de l'hôte comme base d'attaques d'autres réseaux
récupérer, manipuler ou détruire des données
transmettre l'hôte à un ami ou groupe de hackers
se sauver
Maintenant que nous avons mieux cerné le processus d'attaque des hackers, nous allons voir les différentes solutions afin de minimiser les risques.

B) Les systèmes de détection d'intrusion

Les systèmes de détection d'intrusion (IDS) sont devenus un composant critique des architectures de réseau sécurisées. Cependant, ils ne sont encore envisagés que par très peu de professionnels de la sécurité ou d'administrateurs réseau.
Un IDS est constitué de tout matériel, logiciel ou combinaison des deux qui surveille

les activités malveillantes au sein d'un système ou d'un réseau de systèmes. Les IDS sont comparables à un système d'alarme. Ce dernier repose sur l'installation de détecteurs généralement positionnés aux points courants d'entrées et de sorties. Logiquement, cette stratégie se concentre sur les points de la structure considérés comme les plus faibles et donc les plus vulnérables à une effraction. Dans le cadre de la protection d'un élément de grande valeur, la surveillance la plus performante s'obtient à l'aide de détecteurs sensibles capables de détecter les mouvements ou même les changements de température ou de pression de l'air. Les données collectées au niveau des détecteurs sont ensuite transmises à une personne chargée de déterminer la nature de la menace et de réagir en conséquence. Les IDS reposent sur les mêmes impératifs dans le monde des réseaux. Des détecteurs sont placés sur les points d'entrée susceptibles d'être attaqués. Plus la valeur de la ressource d'informations est grande, plus elle sera surveillée avec des détecteurs de plus en plus sensibles. Comme dans le cas d'un système d'alarme, les IDS ont besoin qu'un opérateur humain analyse les données collectées.
Un IDS est un composant critique dans une stratégie de sécurisation des informations de type « défense en profondeur ». La défense en profondeur est une méthode de protection des ressources d'informations fondée sur une série de mécanismes de défense qui se chevauchent. L'objectif est qu'au cas où une ligne de défense serait franchie, les autres sont présentes pour contrer une attaque.
Pour mettre en oeuvre une défense en profondeur, il faut utiliser des hôtes renforcés, des routeurs sécurisés, placer correctement les pare-feu et installer une batterie complète d'équipements supplémentaires. Un IDS s'insère dans cette infrastructure réseau et y surveille toute activité suspecte. Les apprentis dans le domaine de la détection des intrusions font souvent l'erreur de croire qu'un IDS représente une solution de sécurité totale. C'est avant tout un système d'alarme.
Les IDS sont le seul moyen de détecter des attaques hostiles et d'y réagir dans un délai raisonnable. Ils permettent d'effectuer une surveillance complète des réseaux modernes, offrant à une organisation un aperçu en temps réel des menaces qui pèsent sur les systèmes d'informations. En l'absence d'IDS, une organisation peut subir des intrusions à répétition sans qu'elles soient détectées.
La technologie des IDS est dite non « invasive ». En étant correctement configurés, ils ne peuvent dégrader ni perturber le fonctionnement des systèmes.

1. Les différents types d'IDS

Les IDS ont maintenant acquis une certaine maturité, et ils se classent essentiellement en deux types: les IDS réseau (NIDS pour Network IDS) et les IDS d'hôte (HIDS pour Host IDS). Les IDS d'hôte résident sur une machine et surveillent toute tentative d'intrusion sur cette machine. Les IDS réseau sont plus courants car ils surveillent le trafic qui transite sur un réseau à destination d'autres hôtes. Un type n'est pas meilleur que l'autre car chacun est approprié à une situation

spécifique.

a) Les IDS d'hôte

Les HIDS surveillent les attaques perpétrées au niveau du système d'exploitation, d'une application ou du noyau. Ils ont accès à des journaux d'audit, à des messages d'erreur, aux droits de service et d'application et à toute ressource disponible sur l'hôte surveillé. Les HIDS peuvent aussi prendre en compte les applications. Ils savent distinguer des données d'application normales de données anormales. Ils sont capables de surveiller ces données d'application pendant leur décodage et les manipulations dans l'application réelle. Le bénéfice de l'utilisation d'un HIDS est directement lié à cet accès privilégié à l'hôte.
Les HIDS sont particulièrement efficaces pour déterminer si une attaque a réussi. Le trafic malveillant étant fort peu différent du trafic normal, les NIDS sont réputés pour émettre de fausses alertes. De leur côté, les HIDS sont plus performants afin de détecter les intrusions authentiques car ils ne génèrent pas le même volume de fausses alertes qu'un NIDS.
Les HIDS s'appuient sur leur accès privilégié pour surveiller certains composants spécifiques d'un hôte, qui ne sont pas directement accessibles aux autres systèmes. On peut, par exemple, contrôler la bonne utilisation de composants spécifiques des systèmes d'exploitation, comme les fichiers de mots de passe dans Unix et le registre de Windows.
Les HIDS sont réglés sur l'hôte qui les héberge. Ils disposent d'informations nombreuses et précises disponibles uniquement pour un IDS hébergé sur l'ordinateur sous surveillance. Les HIDS peuvent ainsi avoir une connaissance bien spécifique concernant l'hôte et le type d'activité considérée comme normale pour ce dernier. Un flux de données envoyé à l'hôte pourrait apparaître parfaitement normal pour un NIDS, mais être identifié comme anormal et malveillant par un HIDS. C'est pourquoi les HIDS sont capables de détecter certaines attaques que les NIDS laisseront passer.
Les IDS d'hôte présentent des inconvénients majeurs. Puisqu'ils se situent sur l'hôte surveillé, ils ont une vue limitée de la topologie réseau complète. Les HIDS ne sont pas en mesure de détecter une attaque dirigée sur un hôte non équipé d'un HIDS. Un intrus peut compromettre une machine non protégée par un HIDS puis se servir d'un accès légitime vers une machine protégée et le HIDS ne serait pas plus avancé. Pour surveiller les tentatives d'intrusion, le HIDS doit être placé sur chaque hôte critique. Les HIDS s'appuient sur l'hôte pour faciliter la communication avec l'analyste des intrusions; toute attaque qui désactive complètement l'hôte ne sera donc pas signalée.

b) Les IDS réseau

Les NIDS (Network IDS) sont placés sur les zones clés de l'infrastructure réseau et surveillent le trafic destiné aux autres hôtes. Les NIDS ont gagné en popularité et

ont supplanté les HIDS. Un NIDS est plus rentable puisqu'il peut protéger une large portion de l'infrastructure réseau à partir d'un seul périphérique. Le NIDS fournit à l'analyste d'intrusion un angle de vue assez large de ce qui se produit à l'intérieur et autour du réseau. La surveillance d'un hôte ou d'un intrus particulier peut facilement être accentuée ou relâchée. Un NIDS peut offrir une meilleure sécurité et être moins sujet aux interruptions qu'un HIDS. Le NIDS doit être installé sur un seul hôte avec une configuration renforcée, lequel supportera uniquement les services dédiées à la détection d'intrusion, ce qui le rend plus difficile à désactiver.
Puisqu'ils ne dépendent pas de la sécurité de l'hôte, les NIDS sont moins sujets à la destruction des preuves que les HIDS. En effet, les NIDS récupèrent les données et les stockent sur une machine différente, ce qui complique la tâche d'un intrus désireux d'effacer les traces de son passage.
Les NIDS présentent cependant des inconvénients inhérents à leur conception. Pour être efficaces, ils doivent être extrêmement performants dans la prise en charge d'une quantité importante du trafic réseau. Lorsque ce trafic augmente de façon exponentielle avec le temps, le NIDS doit être capable de l'absorber et de l'interpréter de façon appropriée.
Actuellement, les NIDS doivent être judicieusement placés et réglés pour éviter les situations qui peuvent entraîner une perte de paquets.
Les NIDS sont aussi vulnérables aux techniques d'évasion IDS. Les hackers ont découvert de nombreuses méthodes pour masquer leurs opérations de sorte qu'elles ne puissent être détectées par le NIDS.

c) Une approche mixte

Les deux modèles de détection d'intrusion peuvent représenter un composant efficace d'une défense en profondeur. En effet, les NIDS possèdent des avantages qui leur permettent de protéger assez bien de grandes parties de l'infrastructure réseau alors qu'un HIDS offre une protection bien réglée pour des hôtes dont la mission est essentielle tel que les serveurs.
La plupart des organisations qui débutent dans le domaine de la détection d'intrusion commencent par l'installation d'un NIDS. Elles installent ensuite graduellement des HIDS sur les hôtes les plus sensibles. Cette méthodologie apporte une couverture de détection d'intrusion complète pour une organisation.

C) Un IDS hybride : Prelude-IDS

1. Historique

Le projet Prelude a commencé en 1998 et avait pour but de créer un outil modulaire de détection d'intrusion réseau composé d'une sonde et d'un Report Server. Lors du Libre Software Meeting 2001, les équipes de Prelude et du projet Trithème (projet indépendant lancé en février 2000) ont décidé de joindre leurs efforts dans le but d'évoluer progressivement vers le développement d'un IDS hybride basé sur la prise en compte de la quasi-totalité des évènements sécurité au niveau réseau (Network-based IDS) et local (Host-based IDS) grâce à des sondes dédiées.

2. Caractéristiques

Prelude-IDS est un système de détection d'intrusions et d'anomalies distribué sous licence GPL. Ils constituent une véritable révolution dans la détection d'intrusions en étant le premier IDS hybride rendant possible l'administration simple, efficace et centralisée de l'ensemble du dispositif de sécurité quel que soit le nombre de composants, leur marque ou leur licence.
Un tel système vient compléter la panoplie des équipements et logiciels de sécurité (serveurs proxy, routeurs filtrants, firewalls...) et offre à l'analyste un outil de contrôle des activités suspectes ou illicites (interne comme externe).
La détection d'intrusion est réalisée par l'analyse du trafic réseau et l'utilisation de signatures d'évènements hostiles et par l'analyse en continue de fichiers de journalisation.
L'architecture de Prelude est modulaire (on peut intégrer ou développer de nouvelles fonctionnalités grâce à des plugins), distribuée (Prelude est une suite de composants autonomes et interactifs constitué de sondes et de managers) et sécurisée (utilisation du support SSL pour l'authentification et le chiffrement des communications). Les sondes (réseaux comme locales) n'effectuent que les opérations de surveillance et de génération d'alertes alors que les managers prennent en charge la gestion des sondes et la journalisation des alertes.

3. Architecture

Le système Prelude-IDS est constitué de 4 composants essentiels :

a) Le Contrôleur (ou manager)

C'est un serveur haute disponibilité qui reçoit les messages provenant des différentes sondes réseaux et locales et les traduit en alertes. Il applique un ordonnancement en fonction du caractère critique et de la provenance des messages d'alertes. C'est cet ordonnancement qui permet d'établir les priorités de traitement.
Un contrôleur Prelude assure également la remontée des tests de connexion échangés avec les sondes réseaux et locales. Ces tests permettent de vérifier la continuité de la communication entre sondes et contrôleurs.
Le contrôleur est peut-être le composant le plus important d'un IDS à base de Prelude car il ne peut y avoir de journalisation possible sans (au moins) un contrôleur

b) La bibliothèque "Libprelude" :

Elle permet la communication sécurisée entre les différentes sondes et le concentrateur. Elle fournit une interface de programmation (API) pour la communication avec les sous systèmes Prelude et la génération d'alertes au format standard IDMEF. Elle automatise des processus de sécurité se déclenchant en cas de panne d'un ou plusieurs composants du système. L'un des objectifs clés de ce mécanisme est de sauvegarder l'intégralité des alertes en cas de panne d'un concentrateur, par exemple.
Elle constitue la brique de base de tout composant Prelude (à l'exception du frontal Web). Cette libraire fournit aux composants Prelude les fonctionnalités suivantes :
- gestion de la connexion entre composants (sondes et managers) notamment le mécanisme de reprise après interruption et de rétablissement automatique de la connexion ;
- gestion du mode de communication entre composants, notamment la prise en charge du chiffrement éventuel et de l'authentification ;
- interface permettant l'intégration de modules additionnels (plugins).

c) Un système de Détection d'Intrusion machine (HIDS) :

En s'interfaçant à tous types d'applications émettant des alertes de sécurité dans un "journal système", la sonde générique Prelude-lml permet la capture et la dichotomie des informations issues de ces journaux afin de les transformer, ensuite, en alerte Prelude-IDS/IDMEF.

d) Un système de Détection d'Intrusion réseau (NIDS) :

Depuis la Version 0.9. de Prelude, Snort® est devenu le NIDS officiel de la

plateforme Prelude-IDS.

e) Les sondes hôtes et réseau

Avant toute installation, il est nécessaire de faire une analyse préalable concernant le positionnement des différentes sondes au sein du réseau.
Les IDS doivent protéger les composants vitaux et critiques du réseau.

i-Positionnement des sondes hôtes (HIDS)

Les sondes hôtes sont capables de préserver efficacement les serveurs constituant le réseau. Ils permettent d'indiquer toute activité anormale perpétrée contre l'hôte qu'ils protègent.

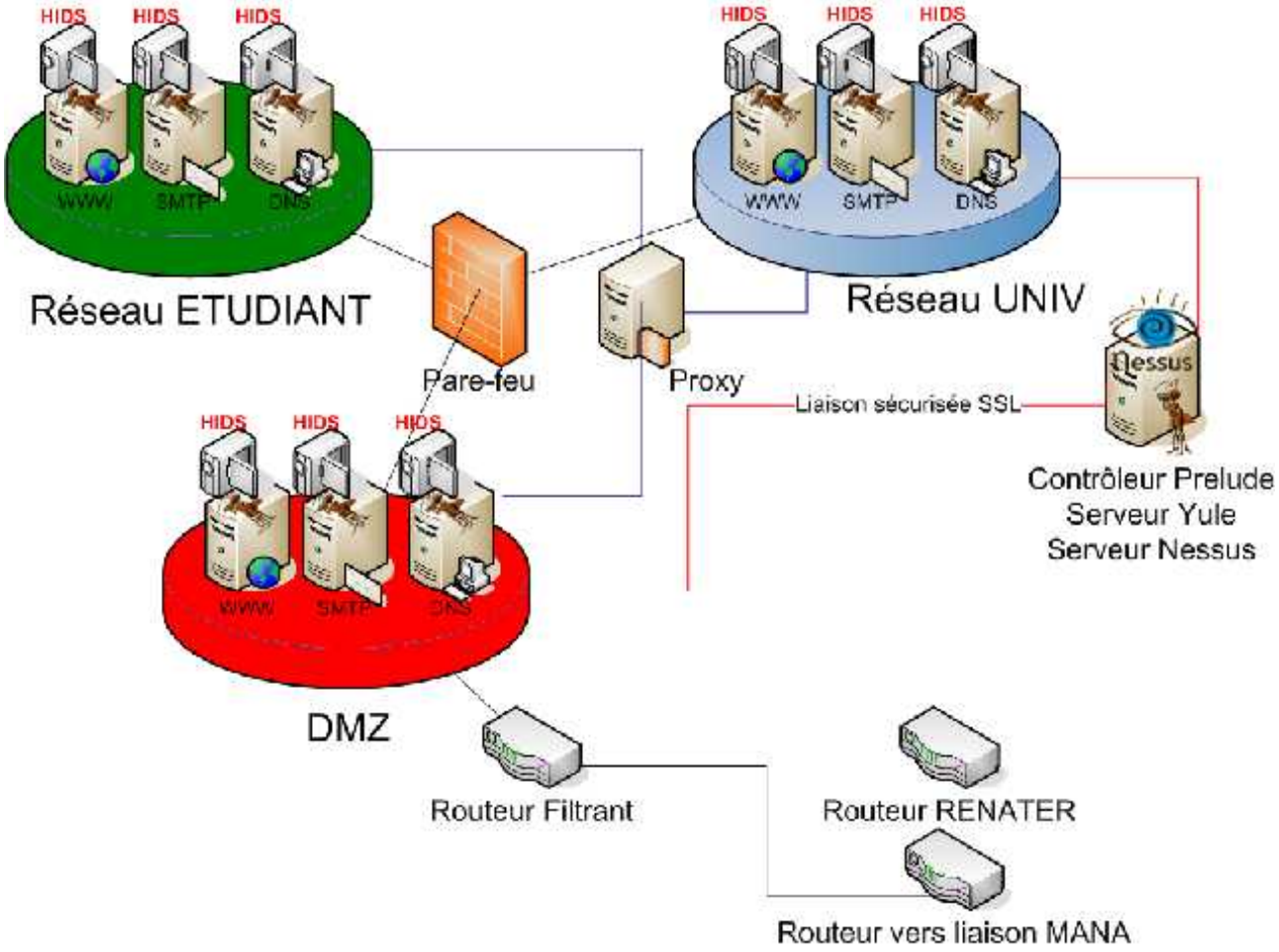


Figure 4: Positionnement des sondes hôtes (HIDS)

ii-Positionnement de la sonde réseau (NIDS) : La DMZ

Lorsque certaines machines du réseau interne ont besoin d'être accessibles de l'extérieur (serveur Web, un serveur de messagerie, un serveur FTP public, etc.), il est souvent nécessaire de créer une nouvelle interface vers un réseau à part,

accessible aussi bien du réseau interne que de l'extérieur, sans pour autant risquer de compromettre la sécurité de l'entreprise. On parle ainsi de « **zone démilitarisée** » (notée **DMZ** pour *DeMilitarized Zone*) pour désigner cette zone isolée hébergeant des applications mises à disposition du public. La DMZ sert ainsi de « zone tampon » entre le réseau à protéger et le réseau hostile.

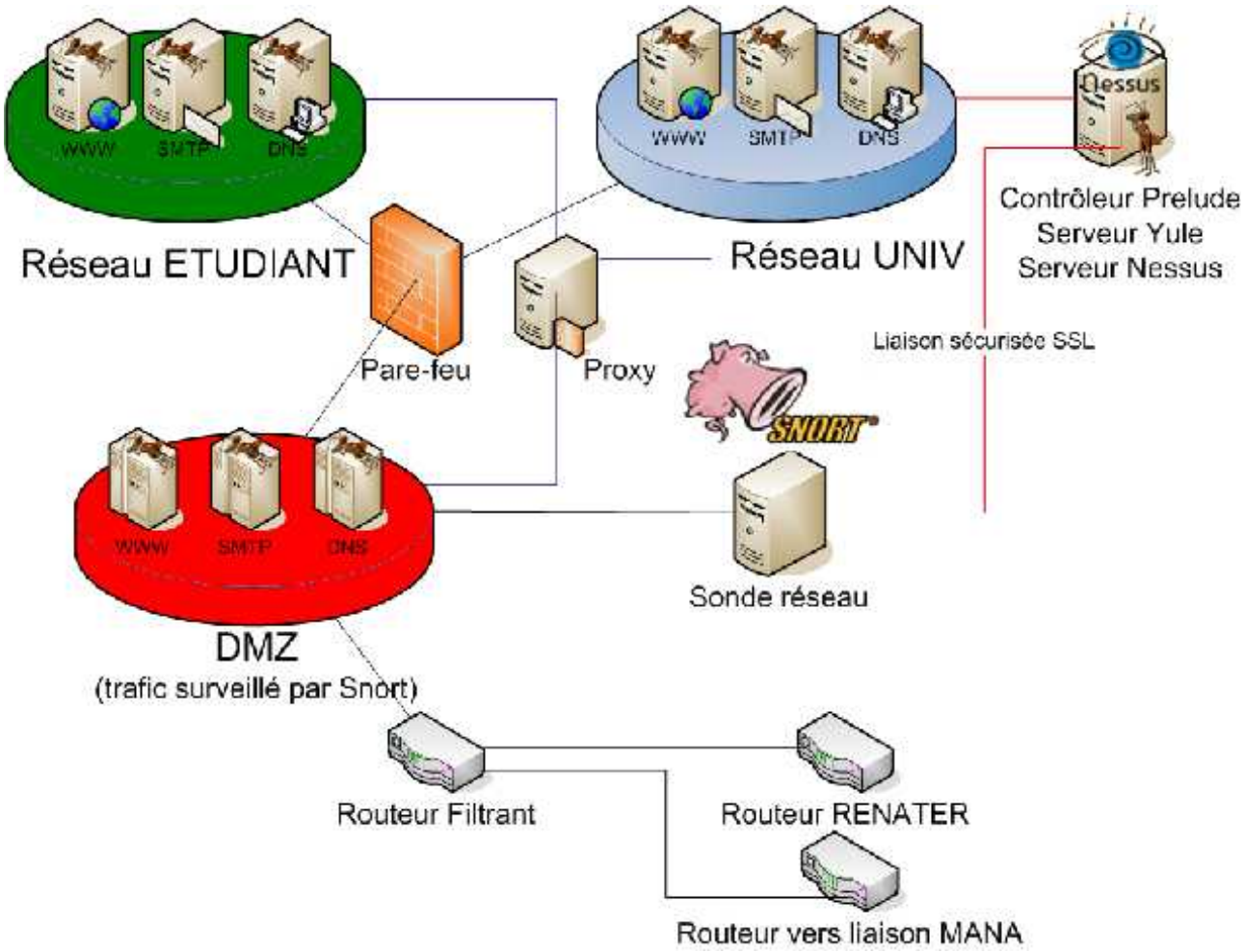


Figure 5: Positionnement de la sonde réseau (NIDS)

Le trafic du réseau externe vers la DMZ étant autorisé, ces machines sont le plus susceptibles de subir une attaque de l'extérieur, c'est pour cela que le choix du positionnement de la sonde réseau s'est portée sur la DMZ afin d'être prévenu en cas de compromission de ces machines.
On pourra envisager, à terme, de mettre en place une sonde sur les réseaux en amont, tel que celui du routeur filtrant vers RENATER ou vers la liaison MANA afin de comparer les résultats et d'en déduire la qualité de filtrage du routeur filtrant.
De plus, on pourrait mettre en place une sonde sur le réseau UNIV et/ou ETUDIANT afin de vérifier le bon usage du réseau notamment concernant l'utilisation de logiciels point à point ou de messagerie.

4. La sonde réseau : Snort

a) Introduction

De simple outil de gestion de réseau, Snort est devenu un système de détection d'intrusion distribué dans les entreprises du monde entier. Depuis sa création en 1998, et après l'installation de presque un demi-million de détecteurs dans le monde, Snort est de loin le NIDS le plus répandu. Son auteur, Marty Roesch, avait initialement conçu Snort comme un outil personnel d'aide à l'analyse du trafic réseau. La version originelle de Snort décodait modestement les données tcpdump binaires pour les afficher dans un format exploitable par l'humain. Roesch avait choisi le nom de son produit au hasard, sans l'intention de le distribuer publiquement et sans imaginer l'immense notoriété qu'il acquerrait par la suite.
Snort a été distribué publiquement et a évolué pendant quelques années pour devenir l'outil incontournable de tout professionnel de la sécurité.

b) Fonctionnement de Snort

Snort capture le trafic destiné aux autres hôtes du même réseau. Snort saisit tous les paquets du réseau et les analysent à l'aide de préprocesseurs afin d'en déterminer la nature bénigne ou malveillante. Snort émet des alertes dès que le trafic semble suspect.

c) Détection de trafic suspect via les signatures

Le moyen le plus efficace aujourd'hui de détecter une tentative d'attaque d'un système ou d'un réseau de systèmes passe par la détection par signature. Cette détection repose sur le principe que du trafic réseau anormal ou malveillant reproduit un modèle spécifique, ce qui n'est pas le cas du trafic normal ou bénin. Le trafic malveillant se distingue en ce qui concerne sa structure et son contenu, c'est pourquoi il est possible de créer une signature d'attaque à partir de laquelle il pourra être reconnu. Snort s'appuie donc sur un ensemble de signatures d'attaque afin de repérer les paquets malveillants.

d) Détection du trafic suspect par l'heuristique

La détection par signatures est très efficace mais ce type de détection ne l'est pas à 100%. Dans certains cas, du trafic peut se révéler dangereux sans exposer de signatures particulières.
La communauté de Snort a développé le module SPADE (Statistical Packet Anomaly Dectection Engine). Cette méthode de détection repose sur la recherche de modèles heuristiques. SPADE observe le trafic réseau et construit une table qui reflète le trafic normal. La table contient des données sur les types de paquets et les adresses de source et de destination. Une fois que la table a atteint une taille significative,

chaque paquet récupéré se voit attribuer un numéro base sur la fréquence à laquelle il apparaît dans la table. Plus le paquet est rare, plus son numéro est grand. Et lorsqu'un seuil configuré est atteint, Snort génère une alerte.

e) Les préprocesseurs

Les développeurs de Snort se sont efforcés de créer une application souple,modulaire pouvant s'adapter aux perpétuelles évolutions des exploitations réseau. Snort possède une architecture modulaire performante qui garantit sa pérennité en tant que système d'intrusion efficace. Il possède une classe de modules d'extension nommés préprocesseurs qui interagissent sur les données avant leur traitement par le moteur de détection. On peut les classer en deux catégories. Ils ont pour rôle soit d'examiner les paquets à la recherche d'une activité suspecte, soit de modifier les paquets de sorte que le moteur de détection puisse les interpréter correctement. Un certain nombre d'attaques ne pouvant être détectées par la recherche de signatures via le moteur de détection, les préprocesseurs d'examen sont appelés en renfort pour détecter les activités suspectes. Les processeurs de ce type sont indispensables pour découvrir les attaques non identifiables par une signature. Les autres préprocesseurs ont pour objectif de normaliser le trafic de sorte que le moteur de détection d'intrusion puisse y rechercher précisément les signatures. Ces préprocesseurs font échouer toute attaque qui tente d'infiltrer le moteur de détection de Snort en manipulant les modèles de trafic.

5. L'interface Web du manager : Prewikka

Prewikka est le nom de l'interface Web permettant un accès unifié à l'ensemble des alertes renvoyées par les sondes. Elle contient divers modules :

a) Système d'Agrégation Avancé

Mise en corrélation d'événements en fonction de critères communs permettant une meilleure lisibilité des attaques et de leurs scénarios

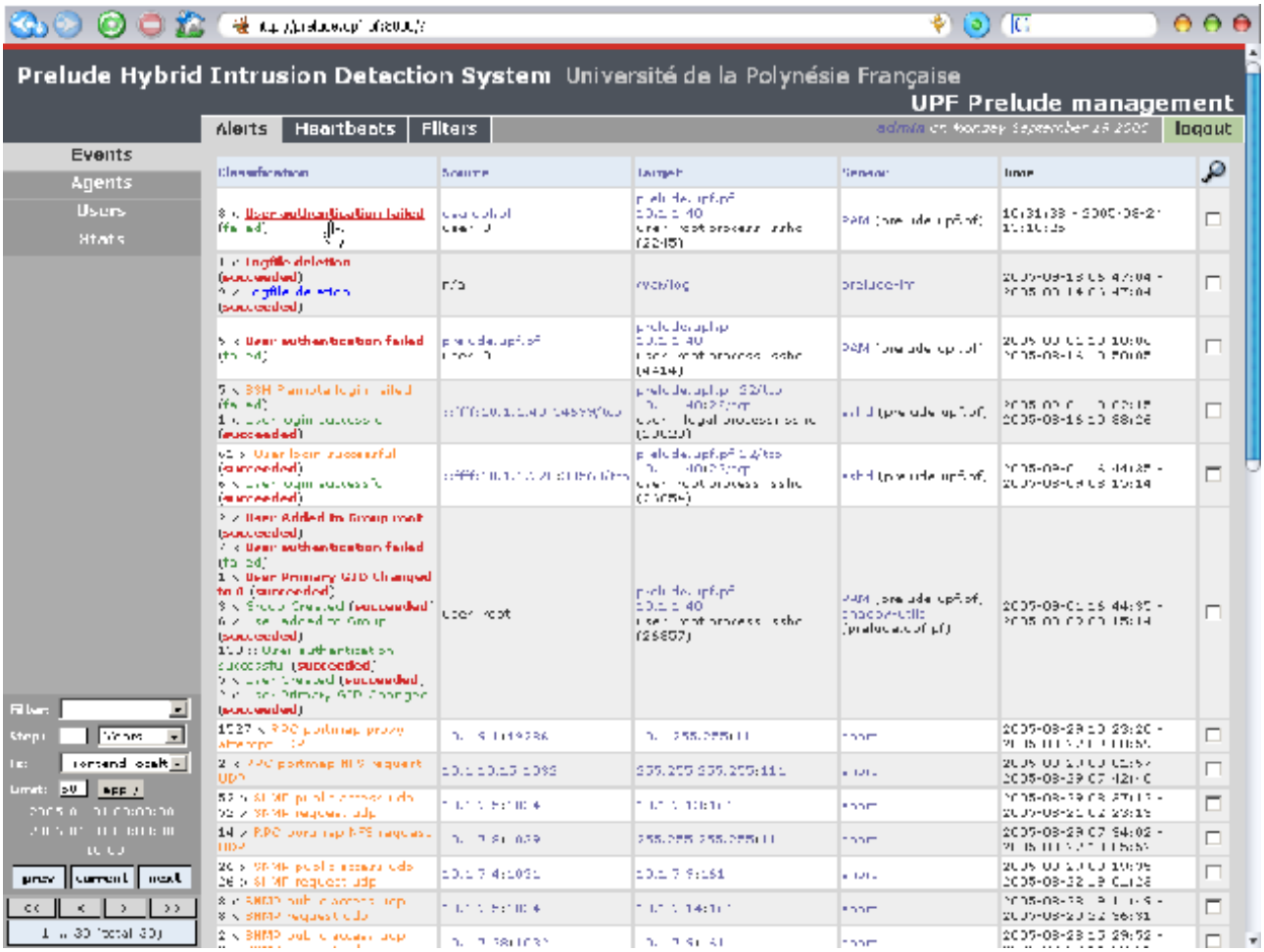


Figure 6: Page d'alertes de l'Interface Web Prewikka

b) Permissions Multi-Utilisateurs

Système de configuration du niveau d'accès des utilisateurs (lecture des événements, modification des événements, gestion des utilisateurs, etc.)

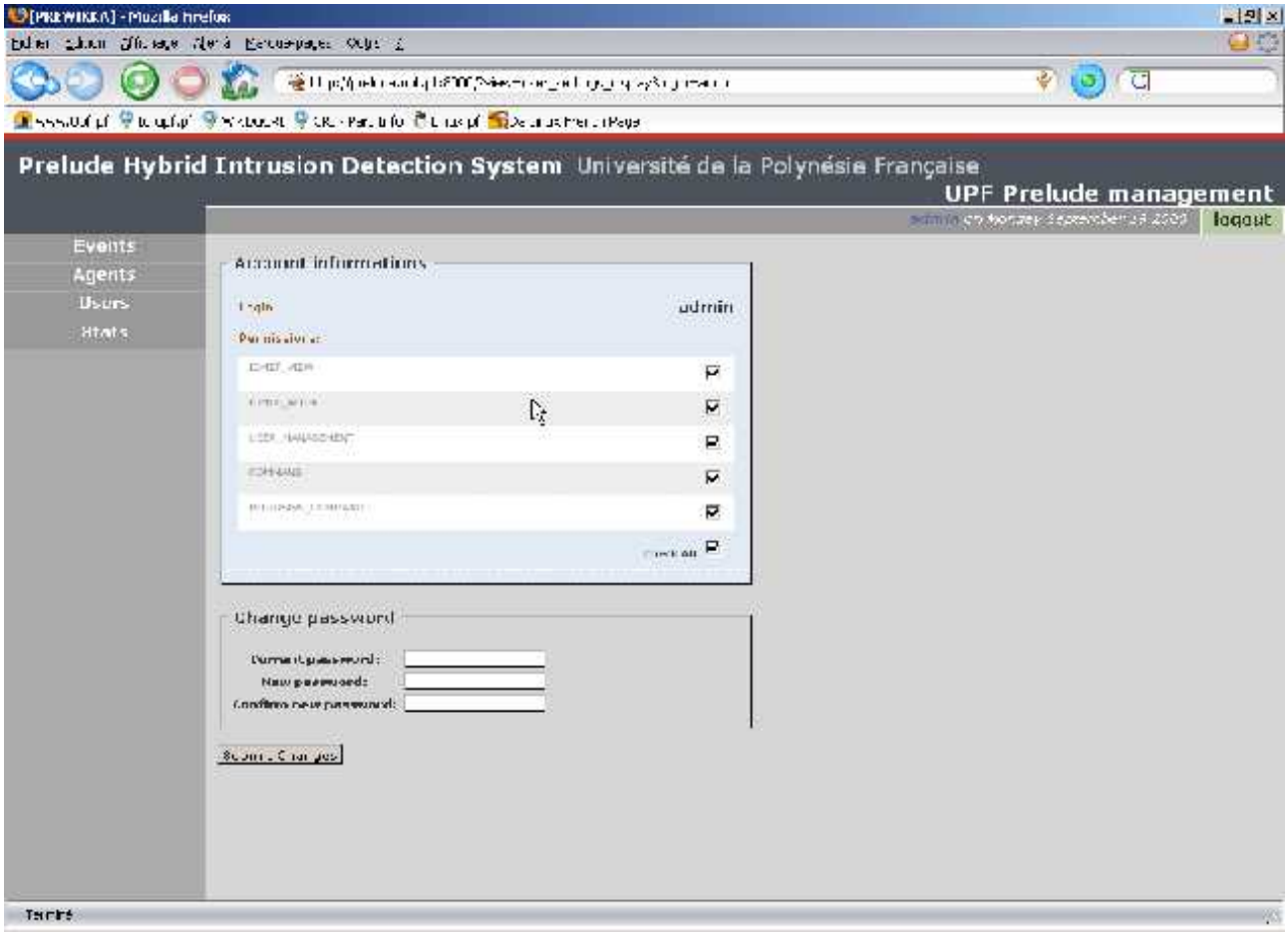


Figure 7: Page des permissions d'accès utilisateurs de l'interface Web Prewikka

c) Création de filtres

Système de définition de critères permettant à l'utilisateur de porter son attention sur des catégories spécifiques d'événements (sonde émettrice, sévérité, complétion, etc.)

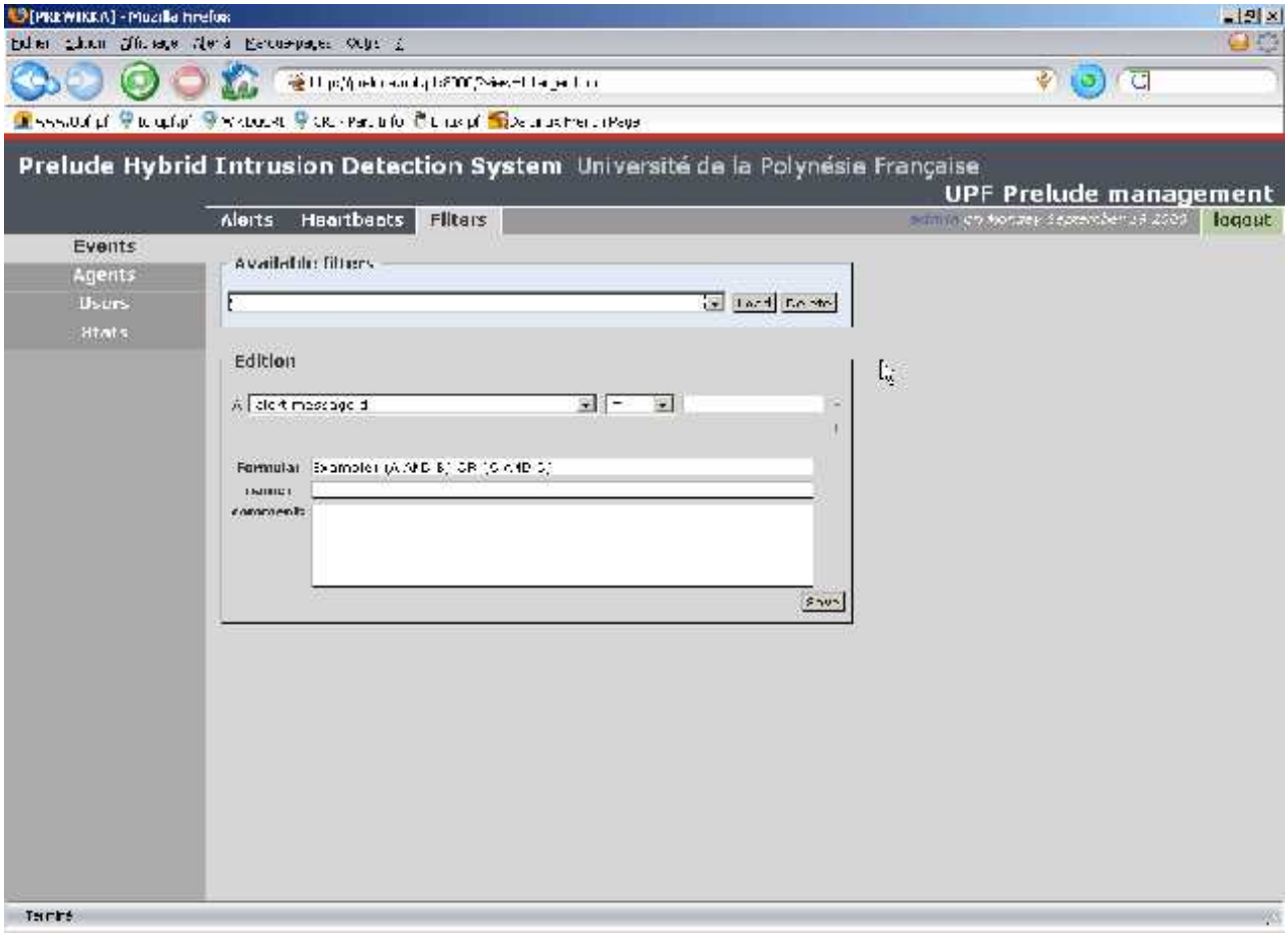


Figure 8: Page de création de filtres de l'interface Web Prewikka

d) Le module des statistiques

Prewikka n'est pas livrée avec le module de génération de statistiques. Celui-ci n'est fournie que dans la version commerciale (accessible pour 3000 €). J'ai donc pris l'initiative de développer, grâce à divers outils tel que matplotlib et pylab, un module de génération de statistiques en python qui permet d'afficher le top 10 des alertes, des sondeurs et des attaquants. Pour le décrire de manière succincte, il récupère les informations de la base de données (requêtes SQL) et les génère sous forme graphique grâce aux fonctions fournies avec les deux outils matplotlib et pylab.
Celui-ci est accessible en annexe.

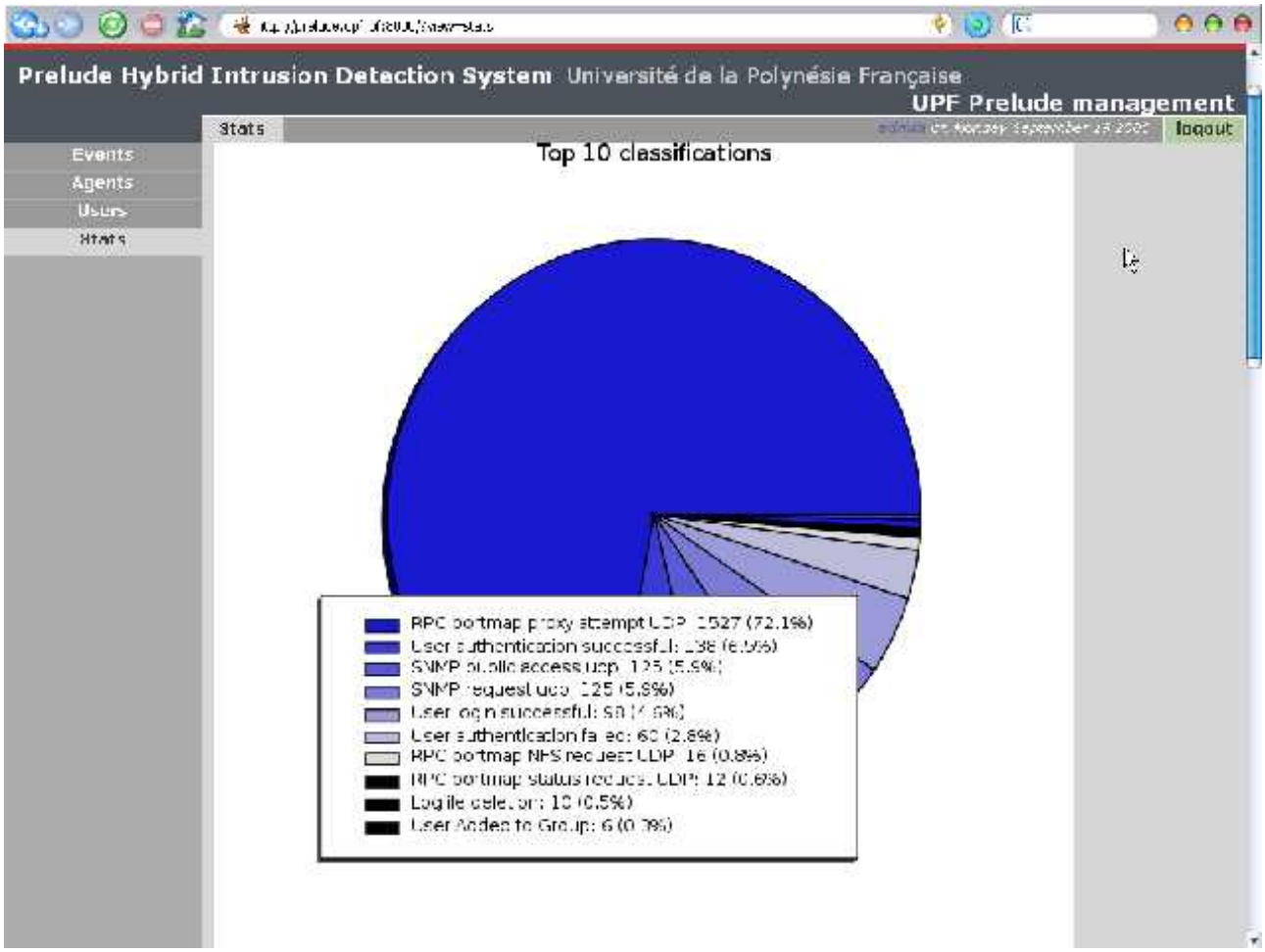


Figure 9: Page des statistiques de l'interface Web Prewikka

6. Intégration d'outils externes

La grande force de Prelude-IDS est de pouvoir intégrer les fonctionnalités d'autres outils de sécurité de référence. On peut, par exemple, utiliser Honeyd (le « pot de miel ») comme une sonde, envoyer les résultats vers le manager qui les intégrera ensuite dans la base de données.

a) Honeyd

i-Présentation

Honeyd est un projet ''Pot de miel'' OpenSource développé par Niels Provos. Il permet de déployer des machines virtuelles sur un réseau (en utilisant les adresses IP laissées libres) et ainsi permet de détecter toute activité frauduleuse sur le réseau. En effet, toute tentative de connexion sur une adresse IP non attribuée peut être considérée comme non autorisée et suspecte. Le but est aussi bien de détecter des attaques connues que de découvrir de nouvelles attaques que d'observer le comportement des attaquants. Ce système constitue une ultime barrière de sécurité.

En effet, l'attaquant sera tenté d'exploiter des serveurs virtuels très vulnérables plutôt que les vrais serveurs qui le sont moins.

ii-Fonctionnement

Honeyd fonctionne sous environnement Unix, Solaris, BSD et sera porté dans l'avenir sous Windows. Les hôtes virtuels générés peuvent être configurés pour qu'ils paraissent fonctionner sous certains systèmes d'exploitations. On peut y simuler certains services (en fait ce sont des scripts qui simulent le fonctionnement de ces services). Honeyd simule des services TCP/IP, peut avoir plusieurs adresses IP (jusqu'à 65536 testées) et supporte ICMP (les machines virtuelles répondent aux ping et aux traceroute). Chaque paquet (entrant et sortant) passe dans un "moteur personnalisé" qui permet de compléter les paquets avec les informations relatives au système d'exploitation simulé.
Plus précisément, Honeyd doit être utilisé en collaboration avec l'outil Arpd. Arpd permet de gérer les adresses IP non attribuées et il redirige les attaques vers Honeyd. Quant à lui, Honeyd gère les échanges de données avec les attaquants pour simuler les services, requêtes ICMP... Sans Arpd, Honeyd ne peut pas travailler.
Il existe tout de même des limitations à l'utilisation de Honeyd puisque peu de services simulés sont disponibles et qu'il ne simule pas tous les systèmes d'exploitation.

7. Bilan

Prelude est un outil intéressant qui permet d'assurer une communication sécurisée entre l'ensemble de ces composants. L'interconnexion de ce système avec d'autres outils est un atout considérable par rapport aux IDS standards. Ce concept d'IDS hybride constitue un outil complet capable d'assurer une prévention fiable sur l'ensemble des composants du réseau.
Le seul inconvénient est l'envoi d'alertes par courrier électronique non disponible nativement. Heureusement, ce module était présent sur un forum pour une ancienne version que j'ai adapté à la version courante. Ce module est fourni en annexe.

III. Le vérificateur d'intégrité : Samhain

A) Introduction

Samhain est un vérificateur libre d'intégrité de données compatible avec Prelude.
Il est un des seuls à fonctionner aussi sous Windows 2000/XP. De plus, contrairement à ses concurrents, il est très rapide car il est écrit en C et n'utilise aucun script interprété.
Samhain est un des seuls à offrir plusieurs dispositifs pour supporter et faciliter le contrôle centralisé. Il peut être utilisé comme un système client/serveur avec le contrôle des clients centralisé et un serveur de log unifié qui rassemble les messages de l'ensemble des clients.
Le fichier de configuration ainsi que les fichiers de base de données peuvent être stockés sur le serveur et être téléchargé par les clients. Le serveur se dénomme yule.
Un de ses avantages est que ce sont les clients qui vont récupérer leur base de données auprès du serveur pour y stocker leurs informations. Si bien que le serveur n'a pas besoin de privilèges root ni de droits d'écriture sur le répertoire dans lequel la base de données est stockée.
Il comprend d'autres fonctionnalités avancées tel que la détection des rootkits, le contrôle des événements de connexion/déconnexion ainsi que la vérification des noms de fichiers anormaux.

1. Qu'est-ce qu'un rootkit ?

Un rootkit est un jeu de programmes installés pour "créer une backdoor (porte dérobée)" après qu'un intrus ait obtenu un accès au système. D'habitude un tel rootkit est très facile à installer et fournit des outils pour cacher l'intrusion (efface par exemple toutes les traces des fichiers de log, installe une commande ps modifiée afin qu'elle n'indique plus certains programmes, etc).
Tandis que certains rootkits "normaux" peuvent être détecté en contrôlant le checksum des programmes (car le 'ps' modifié aura un checksum différent de l'original), d'autres rootkits inversent le processus en modifiant le kernel, ou en chargeant un module du noyau (LKM) (cela permet 'de corriger' le noyau en marche).
Les rootkits de noyau peuvent modifier l'action des appels systèmes. Au niveau utilisateur, ces appels constituent le niveau le plus bas des fonctions systèmes et fournissent l'accès aux systèmes de fichiers, connexions réseau et autres. En les modifiant, le rootkit peut cacher des fichiers, des répertoires, des processus, ou des connexions réseaux sans modifier les fichiers binaires du système. Évidemment, les checksums sont inutiles en pareille situation.

B) Ajustement

Samhain vient avec des fichiers de configuration par défaut pour plusieurs systèmes d'exploitation. Cependant, tous ces fichiers de configuration sont génériques. On peut ajuster des paramètres comme :
. Quels fichiers/répertoires devraient être vérifiés
. Quel type d'enregistrement devrait être utilisé

C) Initialiser la base de données

Samhain travail en comparant l'état présent du système de fichiers par rapport à la base de données de référence. La base de données de référence doit donc être initialisée lorsque le système est sein.

D) Exécuter samhain

Après initialisation de la base de données de référence, vous pouvez exécuter samhain en mode de contrôle.
Vous pouvez utiliser directement le script fourni :
sh$ /etc/init.d/samhain start

E) Amélioration de la régularité des signaux

Afin de minimiser le nombre de fausses alertes, on a besoin de connaître quels fichiers doivent être vérifiés ou non et paramétrer le fichier de configuration en conséquence.
Comme samhain s'exécute en mode démon, il est capable 'de se rappeler' tous les changements du gestionnaire de fichiers, ainsi vous n'aurez jamais deux fois la même alerte.

F) Courrier électronique

Samhain utilise un code SMTP intégré plutôt qu'un diffuseur de messagerie, parce qu'en cas d'échecs de connexion provisoires, le diffuseur de messagerie du système mettrait en file d'attente le message sur le disque, qui pourraient devenir visible aux personnes non autorisées.
Pendant des échecs de connexion provisoires, les messages sont stockés dans la mémoire. Samhain réessayera de les expédier toutes les heures pendant 48 heures.

G) Génération des checksums

Pour les sommes de contrôle de fichiers, Samhain utilise une fonction de hachage particulière, fonction développée par Ross Anderson et Eli Biham. La sortie de contrôle de cette fonction est de 192 bits de long, cette fonctionnalité peut être

mise en oeuvre efficacement sur des machines 32 bits et 64 bits.
Des détails techniques peuvent être trouvés à cette page :
(http: //www.cs.technion.ac.il/~ biham/Reports/Tiger/).
À partir de la version 1.2.10, le MD5 et les fonctions de hachage SHA-1 sont disponibles (on peut les mettre en place en indiquant l'option DigestAlgo=MD5 ou DigestAlgo=SHA1 dans le fichier de configuration). MD5 est un peu plus rapide, mais à cause de soucis de sécurité, il n'est pas recommandé.

H) La définition des fichiers/répertoires à contrôler

Cette section explique comment spécifier dans le fichier de configuration, quels fichiers ou répertoires devraient être contrôlé et quelle politique doit être utilisée.

1. Politique

Samhain offre plusieurs politique prédéfinie de contrôle. Chacune de ces politiques a sa propre section dans Le fichier de configuration.
Les politiques disponibles sont :
ReadOnly
Toutes les modifications sauf les temps d'accès seront reportées pour ces fichiers.
LogFiles
Les modifications d'horodatages, la taille de fichier et la signature seront ignorées.
GrowingLogFiles
Les modifications d'horodatages et la signature seront ignorées. La modification de la taille de fichier sera ignorée si la taille du fichier a augmenté.
Attributes
Seulement les modifications de propriétaire et les autorisations d'accès seront vérifiées.
IgnoreAll
Aucune modification ne sera annoncée. Cependant, l'existence du fichier indiqué ou du répertoire sera toujours vérifiée.
IgnoreNone
Toutes les modifications, y compris le temps d'accès, seront reportées

2. Vérification des noms de fichier anormaux

Samhain vérifie les fichiers ayant des noms anormaux (contenant des caractères de contrôle non imprimable, des retours à la ligne ou des étiquettes) et avertit l'utilisateur.

3. Contrôle des événements login/logout

Samhain peut être compilé pour contrôler les événements de connexion/déconnexion des utilisateurs du système.

I) Yule, le serveur de log

Chaque client potentiel doit être enregistré avec yule pour établir une connexion.
A la première connexion faite par le client, un protocole d'identification est exécuté. Ce protocole fournit l'authentification mutuelle du client et du serveur avec une clé de session.
Par défaut, tous les messages sont cryptés en utilisant Rijndael (choisi comme la Norme de Cryptage Avancée de AES). Une clef de 192 bits est utilisée. Il existe une limite sur le nombre maximal de connexions simultanées. Cette limite dépend du système mais est environ égal à 1000.
La clé de session expire après deux heures. Si la clé de session est expirée, le client est forcé de répéter le protocole d'identification afin de mettre en place une nouvelle clé de session.
Les messages entrants sont signés par le client. À la réception, yule est chargé de :
1. Vérifier la signature,
2. Accepter le message si la signature peut être vérifiée, y renoncer autrement et publier un message d'erreur
4. Enregistrer le message et le nom d'hôte du client dans le fichier de log
5. Ajouter sa propre signature dans le fichier de log.

Il y a un certain nombre d'événements prédéterminés qui peuvent arriver pour un client :
Inactif
Le client ne s'est pas connecté depuis le démarrage du serveur.
Started
Le client a démarré.
Exited
Le client s'est déconnecté
Message
Le client a envoyé un message.
File transfer
Le client est allé chercher un fichier du serveur.
ILLEGAL
Démarrage sans sortie antérieure (peut indiquer une terminaison incorrecte)
PANIC
Le client a rencontré une condition d'erreur fatale.
FAILED
Une tentative échouée d'initialiser une clé de session ou de transférer un message.
POLICY
Le client a découvert une violation de politique.
TIME_EXCEEDED

Aucun message n'a été reçu du client pour un temps défini (par défaut 1 jour, option SetClientTimeLimit).

Le serveur peut manipuler environ 100 connexions par seconde sur un i686 500Mhz. Selon le type d'enregistrement que vous souhaitez, cela devrait suffire même pour un millier de clients.

J) Système de déploiement

1. Méthode A

Samhain inclut un système pour faciliter le déploiement du client aux hôtes distants. Ce système permet de :
construire et stocker des paquetages personnalisés pour des systèmes d'exploitation différents, les installer, créer les bases de données à l'installation, mettre à jour la configuration serveur, initialiser la connexion au serveur et maintenir la base de données des clients nécessaire à l'interface Web beltane.
Le système comprend un script deploy.sh qui est installé dans le même répertoire que yule (par défaut, /usr/local/sbin) ainsi qu'une arborescence de répertoire qui sera installé dans le répertoire de donnée.

2. Méthode B

Samhain fournit une méthode simple afin de créer des paquetages binaires personnalisés avec le gestionnaire de paquetage natif de votre système d'exploitation. Pour se faire :
Bash$./configure [vos options préférées]
Bash$make rpm|deb|tbz2|solaris-pkg
Le paquetage binaire sera alors construit avec les options de compilation choisies du précédent ./configure. Les formats de paquetages supportés sont : rpm (par exemple. Redhat, SuSE...), Deb (Debian), tbz2 (Gentoo Linux) et solaris-pkg (Solaris).
Le paquetage crée utilisera samhainrc de votre système d'exploitation qui se trouve dans le répertoire source, ainsi si vous le personnalisez, votre paquetage contiendra votre version personnalisée.
A l'installation, le paquetage n'initialisera pas automatiquement la base de données de référence et ne lancera pas le démon.

Conclusion

Les réseaux d'entreprises nécessitent un contrôle important de la part des administrateurs réseaux de manière à minimiser les risques. La plupart des administrateurs réseau tentent de protéger leur réseau des attaques extérieures en installant des pare-feu extrêmement efficaces. Malheureusement, à l'inverse, ils n'ont pas autant de préoccupations en ce qui concerne leur réseau intérieur en voulant, légitimement, laisser un maximum de libertés aux utilisateurs.
Le réseau interne est d'autant plus sensible qu'il est possible d'infiltrer plus aisément les serveurs et, ainsi, d'amplifier la portée de l'attaque. En conséquence la plupart des pirates qui veulent arriver à leur fin attaquent la où le réseau est le plus vulnérable : l'infrastructure interne.

Ce procédé peut s'avérer d'autant plus désastreux lorsque les machines constituant leur réseau interne ne sont pas correctement mises à jour. Il sera alors d'autant plus facile pour un pirate d'exploiter des failles déjà connues afin de compromettre le réseau. C'est pour palier à ce problème qu'il existe des outils d'audit tel que Nessus qui permettent d'élaborer des rapports complets concernant le niveau de vulnérabilité du réseau analysé. C'est grâce à ce type d'outil que l'on peut vérifier si les procédés de mises à jour en place sont réellement efficaces.

Néanmoins, même si les ordinateurs sont suffisamment à jour, il existera toujours des failles encore non révélées. C'est ainsi que les pirates prennent au dépourvu les administrateurs réseaux qui n'ont pas jugé bon d'installer des contrôles internes. C'est dans cette politique de prévention qu'interviennent les outils de détections d'intrusions qui permettent de protéger efficacement les serveurs et le réseau en prévenant l'administrateur réseau avant que l'attaque ne se produise.

La mise en place de ces outils nécessite un travail d'analyse préalable afin de générer des résultats les plus pertinents possibles. Ainsi, bien qu'ayant effectué une part de programmation notamment pour la mise en route automatique des serveurs et pour la génération de statistiques, la principale difficulté de mon travail résidait dans la réflexion sur le choix du positionnement de ces outils. J'ai aussi dû effectuer un paramétrage pointu, le plus adapté aux spécificités du réseau de l'Université.

Sur le plan de la programmation, je me serais familiarisé avec le langage python utilisé à la fois pour la génération de statistiques pour l'interface Web Prewikka et l'outil d'envoi automatique d'alertes par courrier électronique.

Ce stage m'aura aussi initié au concept primordial qu'est la sécurité des réseaux. J'aurais pu découvrir l'ensemble des méthodes exploitées par les pirates. Cela m'a permis de me rendre compte d'une réalité certaine : c'est surtout l'inadvertance des administrateurs réseau ou des ingénieurs logiciels qui constitue la

source de toute intrusion. Mais il existe aussi des pirates d'un niveau extrêmement avancé pour lesquels la mise en place de l'ensemble des outils cités constitue, tout de même, une barrière de protection conséquente. Toutefois la sécurité absolue n'existe pas et, dans cette guerre impitoyable livrée par l'ensemble des hackers, spammers et autres crackers, seul les paranoïaques survivent...

Annexes

Table des matières

I. Oscultation d'une attaque

A) Phase de planification

Les hackers planifient généralement à l'avance l'attaque d'un système. Cette planification peut prendre diverses formes. L'attaquant utilise souvent le système d'une façon normale pour l'étudier avant de lancer les hostilités. Il peut s'inscrire pour un compte de courtage sur un système de commerce en ligne ou se connecter à un serveur FTP public. Ce type d'accès légitime ouvert au public lui permet d'établir la portée et les objectifs de l'attaque.
Une fois la préparation initiale achevée, le hacker choisit la portée de l'attaque. Il peut viser différents objectifs, dont les suivants:

- déni de service informatique (Denial of service);
- augmentation des privilèges légitimes
- accès non autorisé
- manipulation de données

Les motivations à l'origine de l'attaque orientent souvent le choix parmi ces objectifs. Un black hat uniquement motivé par la vengeance ou qui veut faire des dégâts choisira une attaque de type déni de service (DoS). Ce type d'attaque procure peu de satisfaction sauf si le hacker tire beaucoup de plaisir de la frustration des autres. Les attaques de type DoS sont très souvent faciles à repérer.

B) Phase de reconnaissance

L'attaquant récupère ensuite des informations ou effectue une reconnaissance de votre réseau. Il va émettre différentes requêtes dont l'objectif est de définir une méthode d'attaque spécifique. La reconnaissance d'un réseau s'effectue de façon analogue à celle du monde physique. Un voleur peut se rendre sur les lieux de son futur forfait pour prendre des photos et noter tous les points d'entrées et de sorties. Il peut se renseigner sur la cible en étudiant toutes les informations disponibles publiquement comme les plans et le calendrier des congés. Il peut même se faire passer pour le propriétaire des lieux et appeler les entreprises de protection et de surveillance pour suspendre le service.

Dans le monde numérique, l'objectif du black hat est de limiter le champ des milliers de possibilités d'actions à un petit nombre de points vulnérables spécifique au réseau visé. L'attaquant tente de réaliser cette reconnaissance en restant aussi discret que possible. Il existe cependant différentes méthodes de reconnaissance et certaines d'entre elles peuvent être détectées à l'aide d'un IDS.

1. Exploiter les données publiques

Bizarrement, il existe de nombreuses sources d'informations accessibles publiquement qui peuvent aider les hackers à compromettre votre réseau. Les bases de données Whois (telles que www.arin.net) et les outils de messagerie électronique publicitaire non sollicitée (tel que www.samspade.org) offrent la possibilité d'identifier les plages d'adresses IP employées par une organisation. Ces outils permettent aussi de savoir si l'organisation héberge ses applications en internes ou si une autre entreprise s'en charge.
Les outils de surveillance publics permettent d'obtenir des informations spécifiques concernant les cibles d'attaque. Le site www.netcraft.com, par exemple, permet d'identifier le système d'exploitation et le serveur Web s'exécutant sur un nom de domaine particulier. La base de données Open Relay, à l'adresse www.ordb.org, permet quant à elle de déterminer si un hôte est vulnérable aux opérations de transmission du courrier électronique.
Une autre méthode courante de récupération d'informations consiste à tirer partir d'un serveur DNS mal configuré. Ce tyoe de serveur détient généralement des informations vitales concernant les hôtes et les relations qui existent entre eux sur votre réseau. Les attaquants tentent souvent un transfert de zone DNS pour obtenir les adresses IP et les noms d'hôte du réseau.

2. Balayer à la recherche de points vulnérables

Une fois qu'un attaquant a fait le tour des sources publiques d'information, il va tenter de détecter les points faibles à exploiter. Il dispose pour cela d'une grande variété de techniques de balayage pour explorer les hôtes.
Les attaquants peuvent tout simplement choisir d'émettre un signal ICMP (ping) à destination des adresses IP afin de déterminer si un hôte écoute à cette adresse. La plupart des infrastructures réseaux externes sont configurées pour ne pas répondre à ce type de requête, cette méthode risque donc d'être peu efficace. L'autre solution consiste à réaliser un balayage de connexions TCP, qui recherche des ports TCP ouverts avec pour objectif de déterminer si l'adresse IP est active.
Dès que l'hôte est détecté à l'adresse IP choisie, l'attaquant recherche alors les ports ouverts via un balayage TCP et UDP complet. Ce balayage permet d'identifier les ports sur lesquels des services se trouvent à l'écoute. Le balayage de connexions TCP est le plus basique. Il consiste à exécuter le processus d'établissement de liaison TCP pour déterminer si un service se trouve à l'écoute. L'attaquant envoie un paquet SYN à l'hôte. S'il reçoit en retour un paquet SYN/ACK, cela signifie que le port est ouvert. S'il reçoit plutôt un paquet RST/ACK, il peut en déduire que ce port est inactif sur l'hôte.
Le balayage UDP est différent à cause de l'orientation « sans connexion » de ce protocole (la transmission s'effectue en une seule opération autonome, sans établir, maintenir ni libérer de connexion). Il n'y a pas d'échange d'accusés de réception

comme dans le cas d'une connexion TCP. Lorsqu'un paquet UDP est émis vers un port UDP de l'hôte qui n'est pas disponible, l'hôte répond qu'il est impossible de se connecter au port ICMP. Si l'attaquant ne reçoit pas cette réponse, il peut supposer que le port est actif. Le balayage UDP est donc moins précis qu'un balayage TCP.
Le balayage du port enregistre aussi toute bannière de promotion des services qui y sont associés. Une bannière est un élément d'information présenté par un service, souvent avant toute authentification. Par ce biais, la bannière de l'hôte peut, s'il est mal configuré, nous préciser son système d'exploitation et le processeur. Il ne serait pas difficile d'exploiter ces informations pour atteindre l'hôte. Une autre méthode moins précise consiste à comparer les ports ouverts avec une liste de ports standard. La plupart des services s'exécutant sur des ports standards bien connus, il suffit à l'attaquant de consulter la liste pour déterminer le type de service présent. En combinant la liste de ports et les bannières, l'attaquant peut facilement deviner quels types de services sont disponibles.
Les hôtes sécurisés sont souvent configurés pour ne pas présenter de bannière et pour s'exécuter sur des ports non standard. Dans ce cas, l'attaquant doit déployer beaucoup plus d'efforts pour déterminer ce qui s'exécute sur ce port particulier. Il doit alors communiquer manuellement avec le service et saisir des commandes factices pour tenter d'obtenir une réponse qui indiquerait la nature du service.
La plupart des exploitations d'infiltration distantes sont spécifiques à un système d'exploitation. L'attaquant doit identifier ce système d'exploitation et sa version pour appliquer la technique appropriée à l'hôte. Il peut alors faire appel à un outil d'identification de système d'exploitation. Cet outil tente d'identifier le système en envoyant divers paquets auxquels chaque système d'exploitation réagit différemment.

C) Phase d'attaques

Après la phase initiale de planification et de reconnaissance, la phase suivante logique consiste à exploiter les informations obtenues et à attaquer le réseau. Le trafic généré dans ce cas peut prendre diverses formes. Tout ce qui ressemble à du code d'exploitation distant en passant par le trafic normal suspect peut signaler une tentative d'attaque qui exige une action.

1. Déni de service

On classe dans la catégorie déni de service (DoS) toute attaque qui perturbe le fonctionnement d'un système de sorte que ses utilisateurs légitimes n'ont plus accès à ces services. Ces attaques peuvent atteindre la plupart des équipements réseau comme les routeurs, les serveurs, les pare-feu, les machines d'accès à distance et la quasi-totalité des autres ressources du réseau. Une attaque DoS peut viser un service spécifique, comme une attaque FTP ou une machine complète. Les types de déni de service sont très diversifiés, mais ils peuvent être classés en deux catégories

: épuisement des ressources et attaques par paquets malveillants.
Les attaques DoS par paquets malveillants consistent à envoyer du trafic anormal vers un hôte jusqu'à ce que le service ou l'hôte lui-même tombe en panne. Ce type d'attaque porte ses fruits lorsque le logiciel n'est pas correctement configuré pour traiter le trafic inhabituel ou anormal. Du trafic hors-norme peut déclencher une réaction inattendue de la part du logiciel, qui tombe alors en panne.
Outre le trafic hors norme, des paquets malveillants peuvent transporter des données capables de faire tomber un système en panne. La charge utile d'un paquet (les données qu'ils transportent) est accueillie en entrée par un service. Si cette entrée n'est pas correctement contrôlée, l'application va subir le DoS.

L'attaque DoS FTP de Microsoft illustre la grande variété d'attaques DoS à la disposition des hackers. La première étape de l'attaque consiste à initialiser une connexion FTP légitime. L'attaquant envoie alors une commande avec une séquence générique (telle que * ou ?). Dans le serveur FTP, la fonction qui traite les séquences génériques dans les commandes n'alloue pas suffisamment de mémoire pour réaliser l'association d'une solution au modèle proposé. La commande d'un hacker comprenant une séquence générique peut ainsi faire tomber en panne le service FTP.
L'autre façon de neutraliser un service consiste à épuiser une ressource. Une attaque DoS par épuisement de ressources consiste à inonder un service avec un trafic normal en grande quantité telle que les utilisateurs légitimes n'obtiennent plus de réponse. Un attaquant qui inonde ainsi un service va saturer les ressources limitées comme la bande passante, la mémoire et les cycles de processeur. Un DoS classique par épuisement de ressources mémoire se nomme SYN-flood. Il tire parti du processus d'établissement de session en trois volets de TCP. Ce processus débute par l'envoi d'un paquet TCP SYN par le client. L'hôte doit alors répondre avec un SYN/ACK. La liaison est établie lorsque le client répond à son tour avec un ACK. Tant que l'hôte ne reçoit pas cette réponse, il attend en conservant la session ouverte. Chaque session ouverte consommant une certaine quantité de mémoire, cette attaque va saturer la mémoire disponible sur le serveur par de multiples sessions TCP inachevées. Le trafic généré par un SYN-flood est normal en apparence. La plupart des serveurs sont cependant configurés aujourd'hui avec un nombre limité de connexions TCP ouvertes.
L'attaque Smurf est une autre attaque classique avec saturation de ressources. Cette attaque profite cette fois des adresses de diffusion réseau ouvertes. Une adresse de diffusion renvoie tous les paquets reçus vers chaque hôte du sous-réseau destinataire. Chacun de ces hôtes répond alors à l'adresse source affichée dans le trafic destiné à l'adresse de diffusion. Lorsqu'un attaquant envoie un flux de requêtes écho ICMP ou de signaux ping sur une adresse de diffusion, le trafic est amplifié jusqu'à 250 fois. Il falsifie aussi l'adresse source de sorte que la cible reçoive l'ensemble du trafic de réponse à l'écho ICMP. Un attaquant disposant d'une connexion Internet ADSL à 128 Ko/s peut facilement générer un flot Smurf de 32

Mo/s.
Les attaques DoS utilisent généralement des adresses IP falsifiées parce qu'elles agissent même si elles ne reçoivent pas la réponse renvoyée. Cette réponse est d'ailleurs inutile, surtout dans le cas de l'attaque Smurf où il est préférable au contraire de ne pas la recevoir. Il est très difficile dans ce contexte de se prémunir d'une attaque DoS et il est d'autant plus délicat d'en détecter l'origine.

2. Les exploitations distantes

Les exploitations distantes sont le moyen le plus pointu d'obtenir un accès non autorisé au système. Les exploitations sont des attaques conçues pour tirer parti de logiciels dont le code est déficient afin de compromettre un hôte vulnérable et d'en prendre le contrôle.
Le fonctionnement des exploitations distantes est analogue à celui des attaques DoS par paquets malveillants abordées précédemment. Ces attaques exploitent les entrées mal contrôlées ou les erreurs de configuration commises par les ingénieurs logiciels.
Une méthode courante repose sur le dépassement de capacité de mémoire tampon (généralement un tableau). Les données se répandent alors dans l'espace d'adressage au-delà des limites de la mémoire tampon. Le résultat se limite généralement à une panne du logiciel. Cependant, lorsque les entrées transmises en entrée sont élaborées d'une façon spécifique, elles peuvent être exécutées de sorte de changer le comportement prévu du système. Cela consiste souvent à générer un shell avec un accès de niveau racine (root). Le dépassement de capacité de la mémoire tampon se produit parce qu'avec l'architecture des machines modernes il n'est pas possible de différencier le code d'une application, des données en entrée.
L'exploitation par codage fragmenté d'Apache est un exemple bien représentatif d'exploitation par dépassement de capacité de mémoire tampon à distance. Lorsqu'il traite des données codées suivant le mécanisme de codage fragmenté, le serveur Apache ne parvient pas à calculer les tailles de mémoire tampon requises à cause de la mauvaise interprétation d'une valeur entière non signée. Des crackers ont appliqué cette technique pour compromettre des serveurs Apache s'exécutant sur diverses plates-formes. Cette exploitation a été la première exploitation distante d'Apache en plus de cinq ans.
Les exploitations distantes peuvent prendre de nombreuses formes qui n'exigent pas forcément une condition de dépassement de capacité de mémoire tampon. Les hackers trouvent souvent le moyen de faire exécuter à une application des commandes arbitraires ou du code binaire sur le système. L'exploitation Unicode du serveur IIS de Microsoft exploite une faille au niveau de la traversée des répertoires. Cette exploitation active une représentation Unicode du séparateur de répertoire (/) pour amener IIS à autoriser un utilisateur à traverser le répertoire principal du serveur Web. L'attaquant obtient l'accès à tous les fichiers du serveur y compris cmd.exe, qui permet d'exécuter n'importe quelle commande DOS.

3. Chevaux de Troie (Trojan) et entrée secrète (Backdoor)

En installant une entrée secrète ou un cheval de Troie, un hacker peut outrepasser les contrôles de sécurité normaux et obtenir un accès privilégié non autorisé à l'hôte. On peut déployer une entrée secrète sur un système de différentes façons. Un ingénieur logiciel mal intentionné peut introduire une entrée secrète dans du code logiciel légitime. Il est aussi possible que ces entrées secrètes soient ajoutées dans le cadre de la maintenance normale au cours du cycle de vie d'un logiciel puis oubliées.
Une attaque de type cheval de Troie est un peu différente puisqu'il s'agit en apparence d'une application ordinaire, sur le principe du cheval de Troie de l'époque gréco-romaine. Un cheval de Troie contrôlé à distance se place typiquement à l'écoute sur un port, comme le ferait une application ordinaire. Par le biais de ce port ouvert, un attaquant les contrôle à distance. On utilise un cheval de Troie pour effectuer toutes sortes d'actions sur l'hôte. Certains chevaux de Troie comprennent des fonctions de balayage de port et DoS. D'autres sont capables de prendre des captures d'écran et de webcam pour les renvoyer à l'attaquant. Les virus de type cheval de Troie et entrée secrète se trouvent typiquement à l'écoute sur un port TCP ou UDP : un responsable de la sécurité peut donc facilement balayer les ports à la recherche des hôtes infectés. Les chevaux de Troie ont évolué récemment et n'ont plus besoin d'écouter un port TCP ou UDP. Les virus de cette nouvelle génération, tels que Sadoor, se placent à l 'écoute d'une séquence spécifique d'événements avant de traiter les commandes : combinaison d'adresses sources prédéterminées, d'informations d'en-tête TCP ou de ports de destination faux qui ne correspondent à aucun service à l'écoute. Les chevaux de Troie emploient d'autres astuces pour déguiser leur présence. Un cheval de Troie courant, Back Orifice, crypte les communications qu'il entretient avec l'attaquant. D'autres chevaux de Troie utilisent des canaux de communication clandestins (tel qu'ICMP). Cette nouvelle race de cheval de Troie donne du fil à retordre aux administrateurs systèmes.

4. Mauvaise utilisation d'un accès légitime

Pour perpétrer son forfait, le black hat peut utiliser un accès légitime ou l'accès d'une autre personne à son insu. On part souvent du principe qu'une personne qui désire endommager une organisation doit passer les contrôles de sécurité. Ce n'est pas vrai : il existe de nombreuses opportunités pour un hacker de détériorer un système tout en utilisant tout simplement un accès légitime.
Les attaquants tentent souvent d'utiliser frauduleusement les comptes légitimes en se procurant des informations d'authentification. Le procédé peut tout simplement consister à se faire passer pour un employé du service d'assistance aux utilisateurs en téléphonant à ces derniers afin de leur demander leur nom d'utilisateur et leur mot de passe. Dans certains cas, cette opération n'est même pas nécessaire : un grand nombre de périphériques sont distribués avec un nom d'utilisateur et un mot de passe par défaut qui souvent ne sont ni supprimés ni modifiés après l'installation. Des

listes complètes de mots de passe par défaut sont proposées sur Internet à l'attention des hackers. Le ver SQL Snakes a démontré le grand nombre de systèmes installés avec des mots de passe par défaut et connectés sur Internet. L'objectif de ce ver était de rechercher les serveurs Microsoft SQL sur lesquels le mot de passe root ou SA par défaut était resté vierge. En quelques heures, le ver a infecté des dizaines de milliers d'hôtes sur Internet. Les attaquants disposent également de méthodes plus avancées pour récupérer des informations d'authentification. Ils font souvent appel à des outils de recherche de mot de passe avec lesquels ils testent automatiquement et à grande vitesse les combinaisons nom d'utilisateur/mot de passe. La méthode utilisée consiste par exemple à tester brutalement toutes les combinaisons de caractères possibles ou à télécharger un fichier dictionnaire avec les noms d'utilisateurs et mots de passe courants. Ce type d'activité de décodage de mots de passe est assez voyant et relativement facile à détecter si une bonne stratégie de sécurité est mise en place.
Même un trafic ordinaire, normal dans des situations suspectes ou inhabituelles, peut signaler l'éventualité d'une intrusions. Si vous remarquez subitement des tentatives d'établissement d'une liaison TCP sur les ports 20 et 21 de votre serveur Web alors que vous n'avez pas de serveur FTP, vous êtes pratiquement assuré que des opérations suspectes sont en cours.

D) Phase postattaque

Si l'attaquant est quelque peu qualifié, il va probablement tenter de masquer ses traces. Il existe plusieurs méthodes pour le faire, dont la plupart impliquent la suppression des preuves et le remplacement de fichiers systèmes par des versions modifiées. Les versions remplacées de ces fichiers ont pour objectif de masquer la présence de l'intrus. Sur une machine Linux, il modifie netstat pour masquer l'écoute du cheval de Troie sur un port particulier. Les hackers masquent aussi leur présence en détruisant des fichiers journaux (log) système ou de sécurité qui auraient pu avertir l'administrateur de leur présence. Ils existent des scripts automatisés qui réalisent toutes ces actions à partir d'une seule commande. Ces scripts sont couramment nommés rootkits.

II. NESSUS

A) Premier compte et démarrage du serveur

Créons un compte d'accès distant. Ce compte nous permettra de nous connecter au serveur depuis notre station de travail (au travers du client Nessus) afin de lancer les contrôles.

1. Création du compte

On utilise la commande nessus-adduser...

```
#/usr/local/sbin/nessus-adduser
        Using /var/tmp as a temporary file holder
        Add a new nessusd user
        ----------------------
        Login : astro
        Authentication (pass/cert) [pass] : pass
        Login password : password
        User rules
        ----------
        nessusd has a rules system which allows you to restrict the hosts
        that upf has the right to test. For instance, you may want
        him to be able to scan his own host only.
        Please see the nessus-adduser(8) man page for the rules syntax
        Enter the rules for this user, and hit ctrl-D once you are done :
        (the user can have an empty rules set)
        ^D
        Login : astro
        Password : robot
        Rules :
        Is that ok ? (y/n) [y] yman page
        user added.
        #
```

Nous avons à présent un compte sur le serveur, avec authentification par mot de passe, dont le login est: astro et le mot de passe: password.

2. Configuration du démon

Le fichier /etc/nessus/nessusd.conf contient les paramètres de configuration du serveur. On peut y indiquer diverses options tel que les plugins à utiliser, leur

emplacement, les options de scan...

3. Création d'un certificat de sécurité

Pour pouvoir démarrer le serveur, il faut créer un certificat.

```
# /usr/local/sbin/nessus-mkcert
    This script will now ask you the relevant information to create the SSL
    certificate of Nessus. Note that this information will *NOT* be sent to
    anybody (everything stays local), but anyone with the ability to connect to your
    Nessus daemon will be able to retrieve this information.
    CA certificate life time in days [1460]:
    Server certificate life time in days [365]:
    Your country (two letter code) [CH]:
    Your state or province name [none]:
    Your location (e.g. town) [Paris]:
    Your organization [Nessus Users United]:
    Congratulations. Your server certificate was properly created.
    /usr/local/etc/nessus/nessusd.conf updated
    The following files were created :
    . Certification authority :
    Certificate = /usr/local/com/nessus/CA/cacert.pem
    Private key = /usr/local/var/nessus/CA/cakey.pem
    . Nessus Server :
    Certificate = /usr/local/com/nessus/CA/servercert.pem
    Private key = /usr/local/var/nessus/CA/serverkey.pem
    Press [ENTER] to exit
    #
```

B) Installation et configuration du client

L'installation de NessusWX (nessus pour Windows) nécessite seulement de décompresser le fichier puis de lancer l'exécutable de l'application dénommé NessusWX.exe.

1. Entrer les paramètres de son compte et se connecter

Allez dans le menu 'Communications' -> 'Connect'. Une fenêtre de dialogue apparaîtra. Entrez le nom (ou l'adresse IP) du serveur, le port (1241 par défaut) ainsi que votre login (astro pour cet exemple) et le type d'authentification (par mot de passe pour cet exemple).

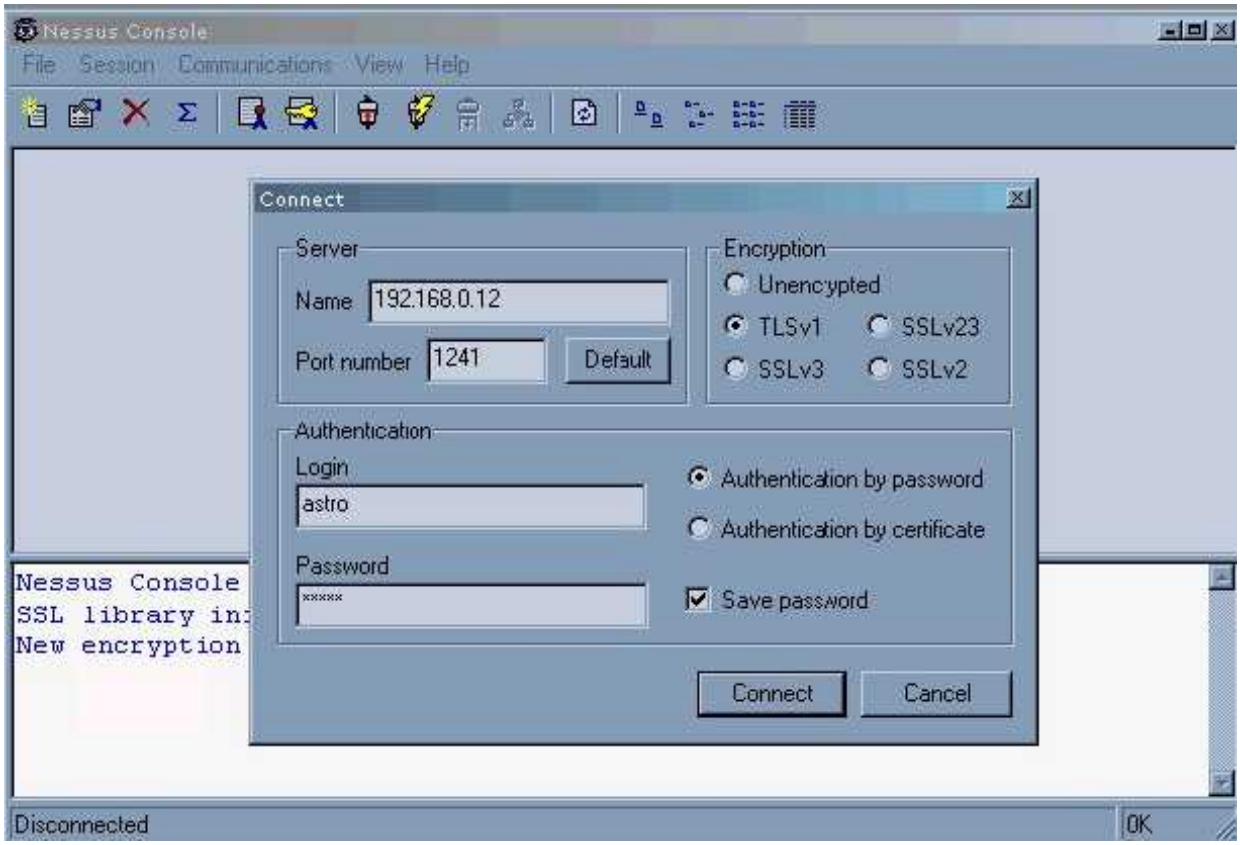


Figure 10: Paramétrage du serveur pour le client Nessus

Cliquez sur 'connect'. Un certificat vous sera proposé, acceptez-le, et entrez votre mot de passe. Vous serez dès lors connecté au serveur.

2. Créer une session de scan

Menu 'Session' -> 'New' , entrez un nom pour la session de scan. Vous aurez alors le panneau de contrôle pour votre session. Ce panneau vous permet de :
1) définir les machines qui vont être scannées
2) Définir les paramètres du scanner de ports
3) Personnaliser les plugins de recherche qui seront utilisés.

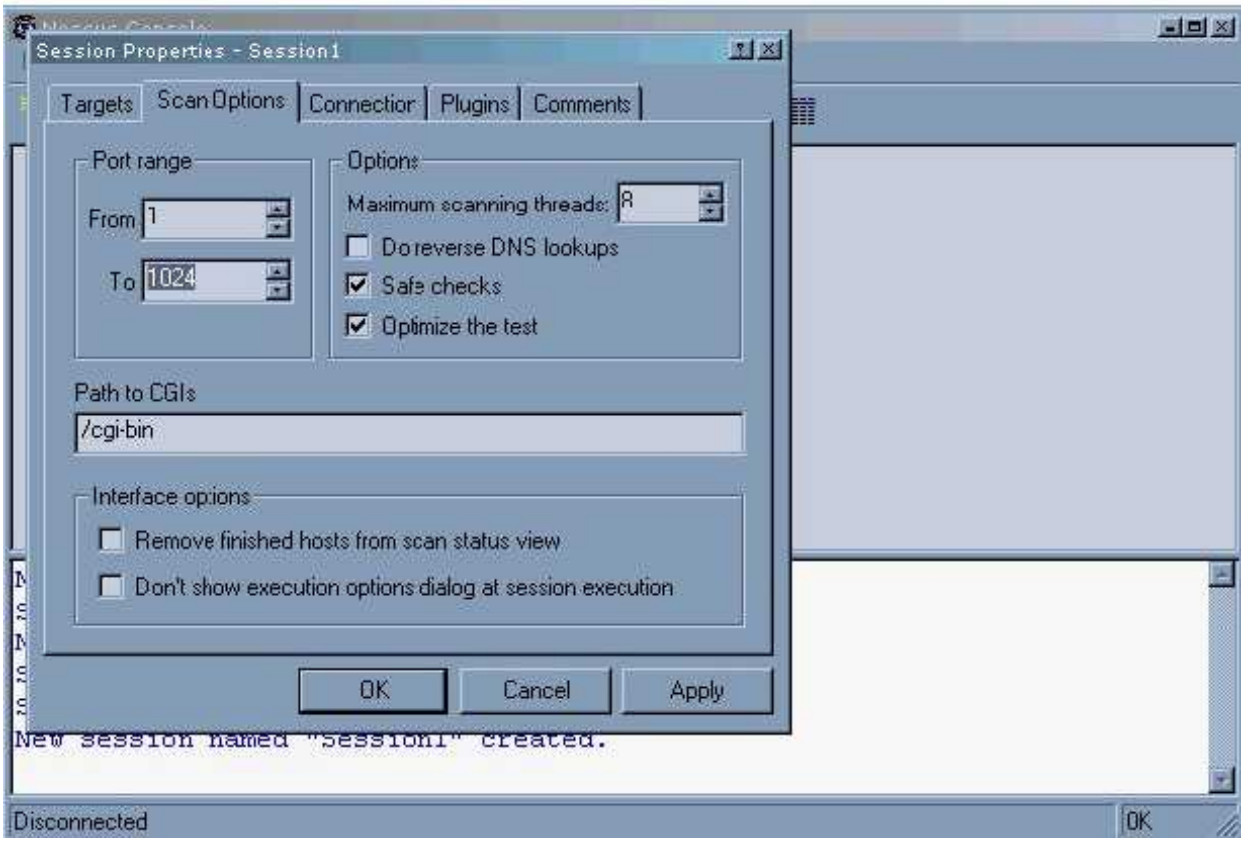
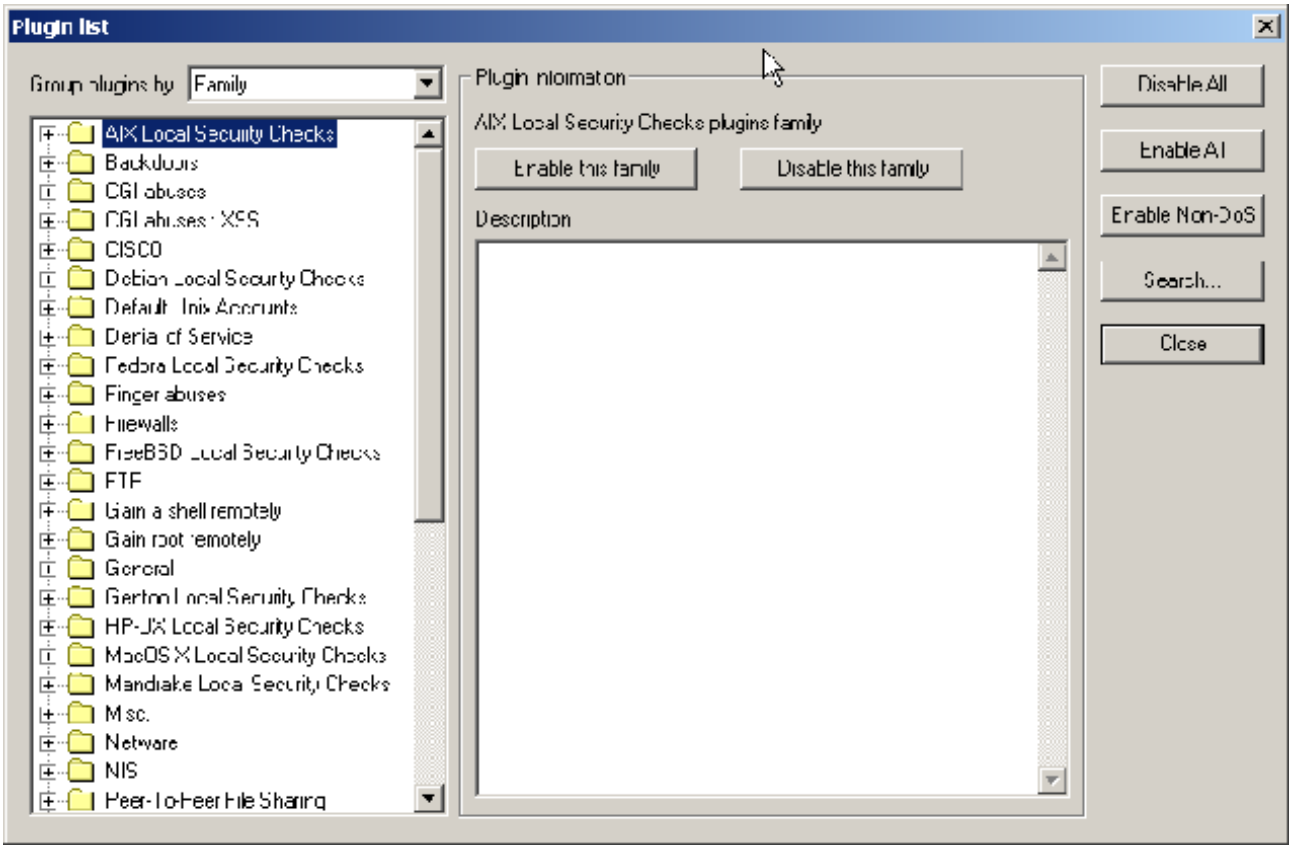


Figure 11: Définition des options de scan pour le client Nessus

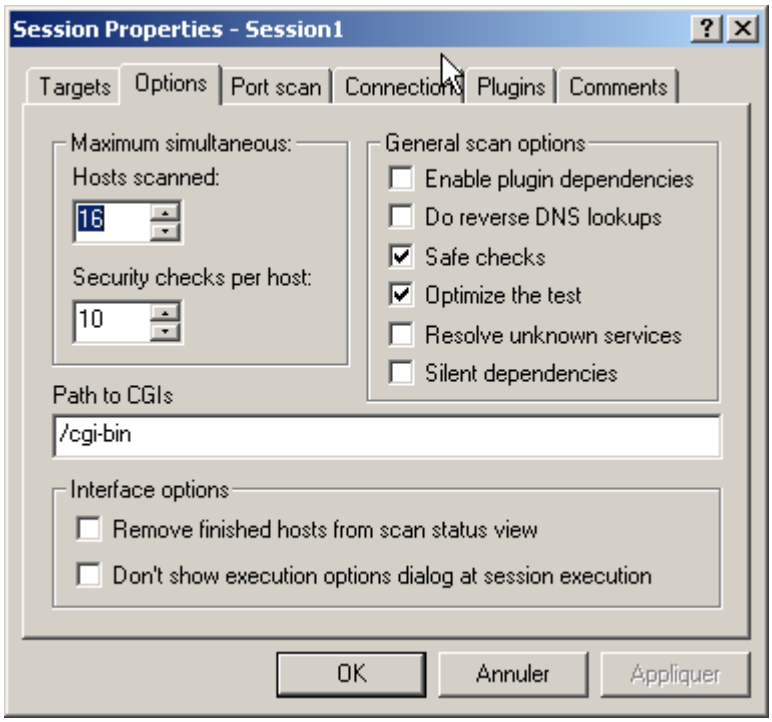
3. Sélection des scripts

L'interface graphique permet de sélectionner les différents scripts voulus (onglet plugins->select plugins), elle permet de :

- sélectionner ou non chaque plugin individuellement.
- sélectionner une famille entière.



4. Options importantes



Trois options dans l'interface influencent la résolution des dépendances :

- Enable dependencies at run time
- Optimize the test
- Consider unscanned ports as closed

Et une quatrième modifie le comportement des scripts agressifs :

- Safe checks

a) Enable dependencies at run time

Par défaut, Nessus ne lance pas un script si ceux dont il dépend n'ont pas été activés. Cette option résout automatiquement les dépendances nécessaires.

b) Optimize the test

Par défaut, Nessus lance même les tests qui n'ont aucune chance de réussir, parce que le service n'a pas été identifié (il attaquera le port par défaut). Cette option accélère le test, au risque de rater quelques vulnérabilités.

c) Consider unscanned ports as closed

Par défaut, Nessus considère les ports non scannés comme "ouverts". Cette option inverse le comportement et accélère le test. Elle n'a d'effet que combinée avec "optimize the test".

d) Safe checks

Cette option désactive les tests dangereux susceptibles de tuer le système ou un service. Nessus s'appuie si possible sur les numéros de version renvoyés par les bannières. Si aucun indice n'est disponible, le test est tout simplement abandonné.
Cette option est moins inoffensive qu'elle paraît :

- Elle peut donner un faux sentiment de sécurité. Qu'une faille ne soit pas mentionnée dans le rapport ne signifie pas qu'elle soit absente du système.
- Si le script est mal écrit et ne vérifie pas la valeur de l'option avec la fonction safe_checks(), l'attaque sera quand même lancé. A priori, les scripts livrés avec Nessus sont sûrs, mais un script "non officiel" ou expérimental pourrait être dangereux.

NB : les scripts ACT_FLOOD, ACT_DENIAL, ACT_KILL_HOST et ACT_DESTRUCTIVE_ATTACK ne sont jamais lancés quand cette option est activée.

Après avoir validé les divers paramètres, un livre apparaîtra sur la fenêtre centrale, il représente l'enregistrement de la session.

5. Lancer le scan

Cliquez avec le bouton droit sur le livre, et vous aurez l'option 'execute', cliquez dessus. Vous accédez alors à quelques options concernant le déroulement du scan.

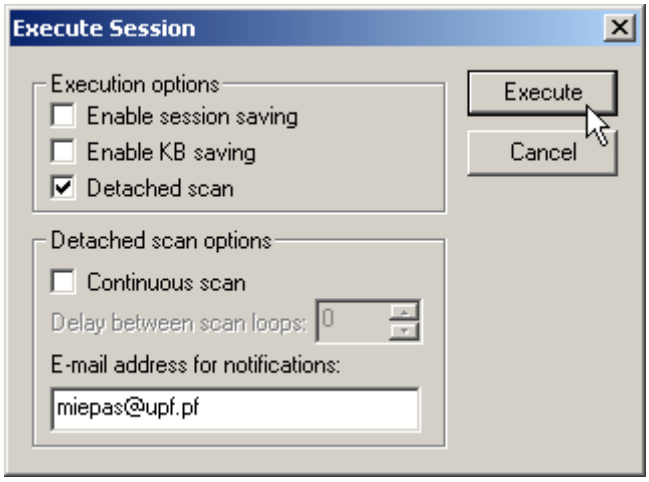


Figure 12: Validation du scan pour le client Nessus

Un scan 'detached' est un scan qui ne sera pas traité par le client, mais par le serveur lui-même. Le serveur vous enverra le bilan du rapport à l'adresse e-mail que vous aurez entrée.

Sachez qu'un scan de ports UDP de 1 à 65000 peut facilement durer plus de 24 heures. Préparez-vous donc à ne pas forcément avoir les informations sur les scans de suite.

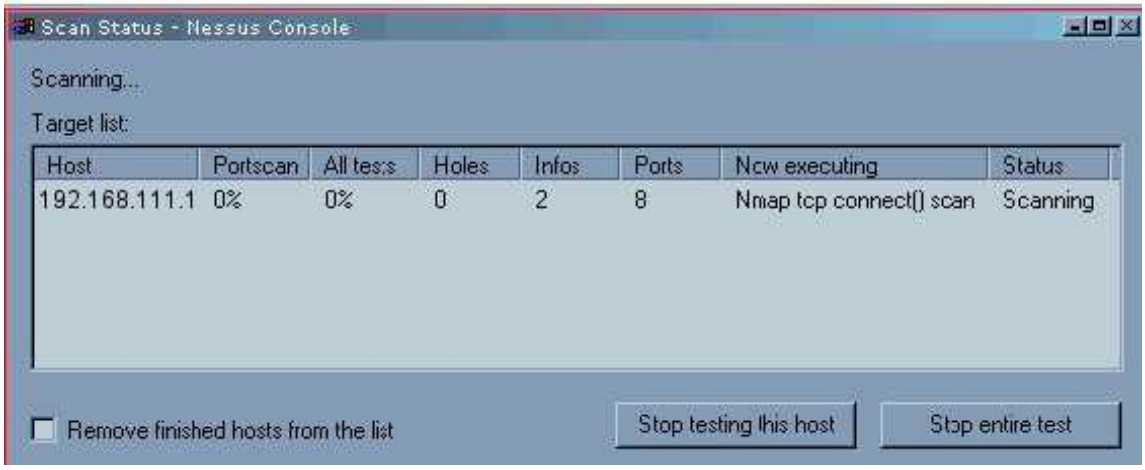


Figure 13: Déroulement du scan pour le client Nessus

6. Le rapport final

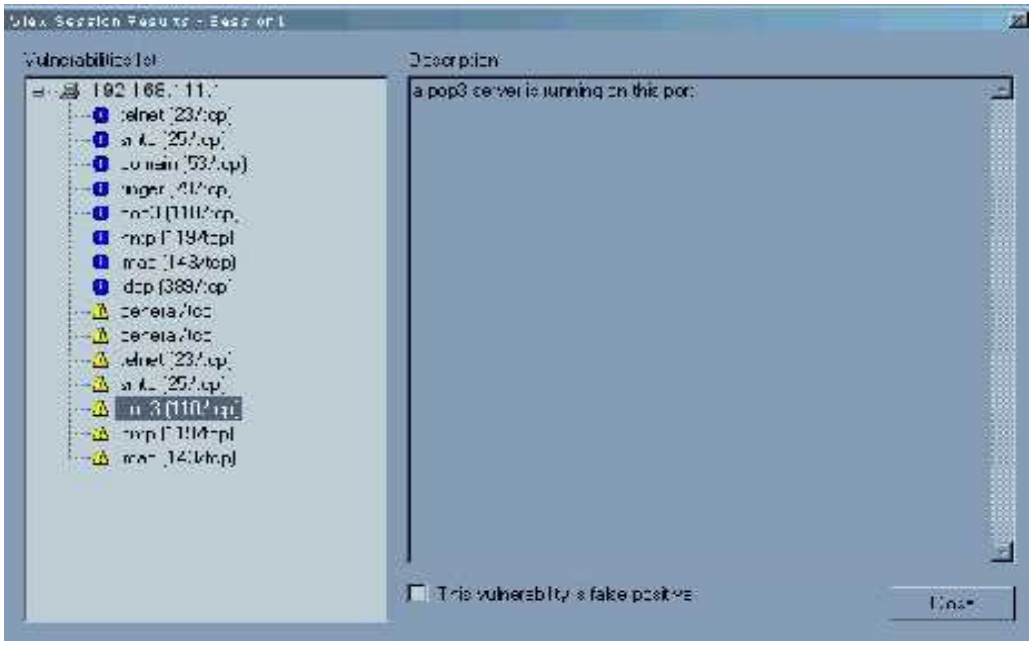


Figure 14: Rapport généré par le client Nessus

A la fin d'un scan, vous accéderez au panneau des vues ('results') en cliquant avec le bouton droit sur votre icône de session. Elles sont classées chronologiquement selon vos demandes de scans, et vous pouvez choisir le type de sortie voulu. N'hésitez pas à les sauvegarder (save) et également à les exporter (export) dans un format tel que le format HTML.

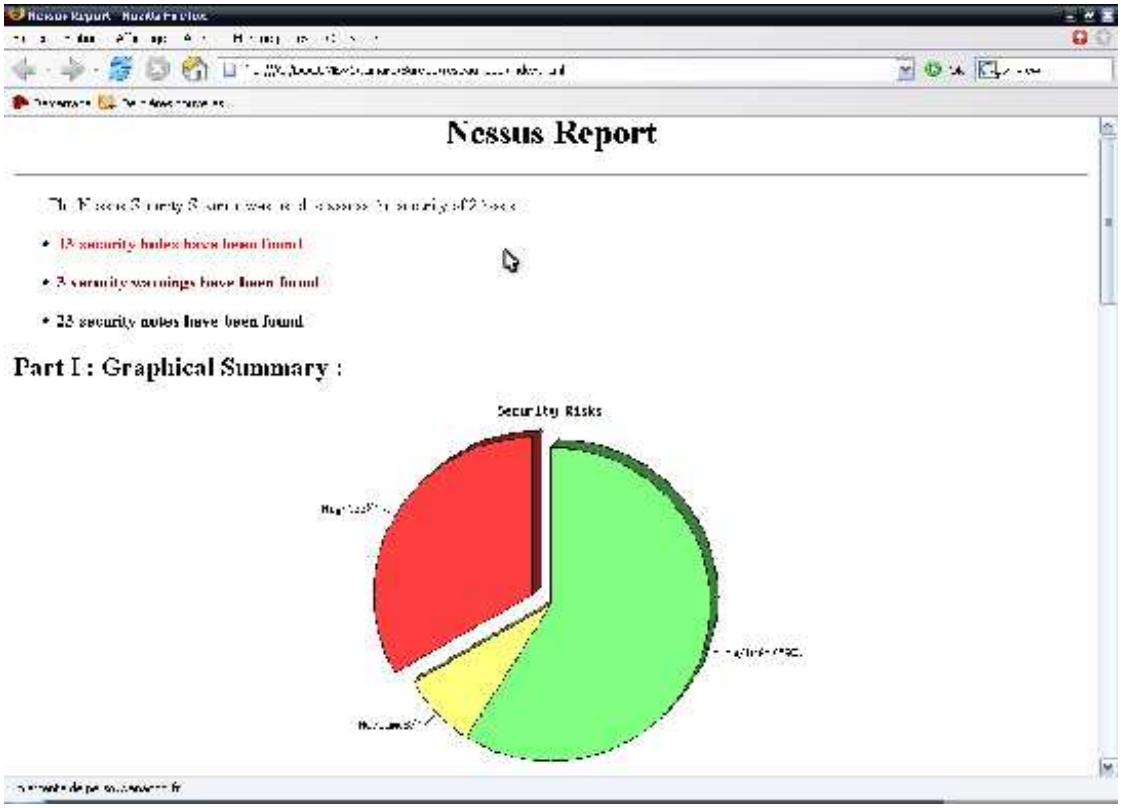


Figure 15: Rapport HTML généré par le client Nessus

III. Installation de Prelude-IDS

Prelude-IDS nécessite un ensemble de modules livrés nativement dans le système debian tel que OpenSSL et gnutls. Ainsi leur installation n'est pas expliquée dans cette section.
L'installation des différents composants Prelude est relativement simple puisqu'elle repose, après extraction de l'archive, sur la désormais célèbre trilogie " configure - make – make install ".

A) Installation de la librairie libprelude

Pour tous types d'installation, que ce soit le concentrateur, la sonde hôte ou la sonde réseau constituant le système Prelude-IDS, il faut au préalable installé la libraire Prelude : Libprelude. Cette libraire constitue le composant essentiel pour la communication entre le concentrateur et les sondes en assurant son chiffrement et donc sa sécurisation.
L'installation de la librairie Prelude n'échappe pas à la règle énoncée ci-dessus :
- configure (par défaut, s'installe sous /usr/local).
- make
- su
- make install

A l'issue de l'installation, les librairies sont installées et un répertoire prelude-sensors a été créé, dans lequel un fichier de configuration par défaut (sensors-default.conf) a été installé.
Durant la phase d'installation, les supports pour les langages Perl et Python requis auront ou non été trouvé. Ces supports constituent une API (Application Program Interface) qui va définir un ensemble d'outils afin de développer des applications pour Prelude par l'intermédiaire de ces deux langages. C'est notamment sous le langage Python que l'interface Web Prewikka est développée.
Un utilitaire prelude-adduser est également présent dans /usr/local/bin. Il sera utilisé pour enregistrer les composants Prelude auprès des contrôleurs.

B) Installation d'un contrôleur

Compte-tenu de l'architecture distribuée de Prelude, il faut commencer le déploiement de l'IDS par l'installation d'un contrôleur (prelude-manager).
Pour l'installation du contrôleur, si l'on veut enregistrer les alertes dans une base de données, il est nécessaire d'installer au préalable, libpreludedb. Cette librairie permet de transmettre les données IDMEF renvoyées par les sondes dans une base de données. Elle nécessite, bien entendu, l'installation d'un système de gestion de base de données (Msql ou Postgresql).

1. Installation de libpreludedb

./configure (par défaut, s'intégre à mysql ou postgresql si détecté)
-make
-su
-make install

2. Installation de prelude-manager

Une fois encore, la procédure est simple :
- configure (par défaut, s'installe sous /usr/local).
- make
- su
- make install

Notez que si vous décidez d'utiliser une base de données (MySQL ou PostgreSQL) Pour journaliser les alertes Prelude dans une base de données, il suffit de lancer le script prelude-manager-db-create.sh qui, comme son nom le laisse deviner, va créer les bases de données nécessaires. Le SGBD étant installer sur le même hôte que le contrôleur garantit que les communications entre les sondes jusqu'à la journalisation sont chiffrées.
En plus du binaire, un fichier de configuration (prelude-manager.conf) est installé dans /usr/local/etc/prelude-manager.
Ce fichier déterminera le comportement du contrôleur :
- adresse et port d'écoute ;
- activation des modules de sorties (fichier texte, xml ou insertions SQL).

L'activation des modules et agents de contre-mesure sera également configurée dans ce fichier.

C) La sonde hôte : Prelude-lml

- configure (par défaut, s'installe sous /usr/local)
- make
- su
- make install

En plus du binaire (que l'on trouvera sous /usr/local/bin), les fichiers de configuration
suivants sont installés sous /usr/local/etc/prelude-lml :
- prelude-lml.conf : définit les paramètres généraux tels que l'adresse des contrôleurs associés à la sonde, les objets (fichiers) à surveiller, et déclare les modules de détection. Cela installe aussi :

un répertoire ruleset contenant les règles (sous forme d'expressions régulières compatibles Perl, PCRE) de filtrage qui déclencheront l'envoi de messages par la sonde. Ces règles sont appliquées sur les fichiers de logs du système afin de pouvoir

recueillir l'ensemble des alertes générées par les programmes installés sur le système.

1. Génération du détecteur

Notre détecteur est constitué de deux cartes réseaux : la première pour écouter le segment de réseau à surveiller (la DMZ) et l'autre pour renvoyer les alertes au serveur Prelude (prelude-manager).
La carte réseau qui est sur la zone démilitarisée est configuré pour s'exécuter en mode discret ce qui signifie qu'aucune adresse IP ne lui est attribuée.
Pour se faire, sous Linux, il suffit de taper :
ifconfig eth1 up
ifconfig eth1 promisc
Avant toute chose, il faut installer libprelude (comme expliqué précédemment), de manière à ce que Snort transmette les alertes au serveur Prelude.
Les sections suivantes expliquent l'installation d'autres composants nécessaires à Snort.
Installer libpcap
Snort ne disposant d'aucune fonctionnalité de capture de paquet native, il doit faire appel à une bibliothèque de reniflage de paquets externes, libpcap. Elle est chargée de récupérer les paquets directement à partir de la carte d'interface réseau. Il faut donc télécharger la dernière version de libpcap, la compiler et l'installer grâce à :
./configure
make
su
make install

a) Installation de Snort

Snort étant compatible, à partir de la version 2.4, avec Prelude, il suffit de l'indiquer au moment de la configuration des paquets.
./configure –enable-prelude
make
su
make install
L'application est maintenant installée. Il faut maintenant créer un dépôt pour les règles et les fichiers de configuration de Snort:
mkdir /etc/snort
chmod 700 /etc/snort
cp etc/* /etc/snort/
Le répertoire /etc/snort contient le fichier de configuration de snort : snort.conf. Il contient divers paramètres importants tel que les plages d'adresses à écouter, les ports, les adresses des serveurs DNS, SMTP, HTTP, SQL... ainsi que les paramètres

des préprocesseurs.
Les signatures présentes dans la distribution source sont les plus récentes au moment de la mise sur le marché de la version. En conséquence, elles sont probablement obsolètes, il est donc nécessaire de télécharger les dernières sur le site officiel de snort. Il faut ensuite les décompresser dans le répertoire /etc/snort.

D) Module python pour la génération des statistiques

Ce module permet de générer les statistiques de Prewikka tel que le top 10 des classifications, des attaquants et des sondes.

```
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2, or (at your option)
# any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; see the file COPYING.  If not, write to
# the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.

import time
import matplotlib
matplotlib.use("Agg")

from prewikka import User, view, utils, siteconfig

from pylab import *

GRADIENT_COLOR_START = (17.0/255, 17.0/255, 191.0/255)
GRADIENT_COLOR_END = (209.0/255, 209.0/255, 209.0/255)
COLOR_BLACK = (0.0, 0.0, 0.0)

def get_color_gradient(steps):
    colors = [ GRADIENT_COLOR_START ]
    if steps == 1:
        return colors
    step = [ (e - s) / (steps - 1) for e, s in zip(GRADIENT_COLOR_END,
GRADIENT_COLOR_START) ]
    prev = GRADIENT_COLOR_START
    for i in range(steps - 1):
        new = [ x + s for x, s in zip(prev, step) ]
        colors.append(new)
        prev = new

    return colors



class Chart:
    def __init__(self, filename):
        #self._filename = "/usr/share/prewikka/htdocs/generated/" + filename
        self._filename = filename
        self._title = ""
        self._values = [ ]
        self._labels = [ ]
```

```
        self._values_title = None
        self._labels_title = None

    def getFilename(self):
        return self._filename

    def setValuesTitle(self, title):
         self._values_title = title

    def setLabelsTitle(self, title):
         self._labels_title = title

    def setTitle(self, title):
        self._title = title

    def setValues(self, values):
        self._values = values

    def setLabels(self, labels):
        self._labels = labels

    def addLabelValuePair(self, label, value):
        self._labels.append(label)
        self._values.append(value)

    #def _render(self):
    #    figure(facecolor="b")
    #    title(self._title)
    #    savefig(self._filename)
    #    clf()


   # def renderPie(self):
    #    total = float(reduce(lambda x, y: x + y, self._values))
     #   patches, texts, autotexts = pie(self._values, labels=self._labels,
autopct=lambda x: "%f" % x)
      #  for autotext, value in zip(autotexts, self._values):
       #     autotext.set_text("%d (%.1f%%)" % (value, value / total * 100))
        #self._render()

    #def renderPlot(self):
     #   plot(self._values)
      #  xticks(arange(len(self._values)), self._labels)
       # self._render()

    def _render(self):
        title(self._title, verticalalignment="center")
        if self._values_title:
              ylabel(self._values_title)
        if self._labels_title:
               xlabel(self._labels_title)
        savefig("/usr/share/prewikka/htdocs/generated/" + self._filename)
        clf()

    def renderPie(self):
        figure(figsize=(7,7))
        total = float(reduce(lambda x, y: x + y, self._values))
        values = [ ]
        colors = get_color_gradient(len(self._values))
        for value in self._values:
            if value / total < 0.007 and len(self._values) - len(values):
                values.append(reduce(lambda x, y: x + y,
self._values[len(values):]))
                colors = get_color_gradient(len(values) - 1)
                colors.append(COLOR_BLACK)
                break
```

```
            values.append(value)
        patches, texts, autotexts = pie(self._values, colors=colors,
autopct=lambda x: "", shadow=True)
        #patches, texts, autotexts = pie(values,
labels=self._labels[:len(values)], colors=colors, autopct=lambda x: "",
shadow=True)
        for text in texts:
            text.set_text("")
        labels = [ "%s: %d (%.1f%%)" % (label, value, value / total * 100) for
label, value in zip(self._labels, self._values) ]
        l = legend(labels, loc=(0,0), shadow=True)
        i = 0
        for patch in l.get_patches():
            if i < len(values):
                color = colors[i]
            else:
                color = colors[-1]
            patch.set_fc(color)
            i += 1
        l.set_alpha(0.75)
        self._render()

    def _setYticks(self):
        locs, labels = yticks()
        step = int(locs[1] / 2)
        ymin, ymax = ylim()
        r = range(int(ymin), int(ymax + step), step or 1)
        yticks(r, r)

    def _setXticks(self):
        xticks(arange(len(self._values)), self._labels,
horizontalalignment="center")

    def _setTicks(self):
        self._setXticks()
        self._setYticks()

    def renderPlot(self):
        plot(self._values)
        self._setTicks()
        self._render()

    def renderBar(self):
        bar(arange(len(self._values)), self._values, color=GRADIENT_COLOR_START)
        self._setTicks()
        self._render()

class Stats(view.View):
    view_name = "stats"
    view_template = "Stats"
    view_permission = [ User.PERM_IDMEF_VIEW ]
    view_parameters = view.Parameters

    def _render_distribution(self, filename, title, path, limit):
        chart = Chart(filename)
        chart.setTitle(title)
        for value, count in self.env.idmef_db.getValues([ path + "/group_by",
                                                           "count(alert.messageid)
/order_desc" ],
                                                         limit=limit):
            chart.addLabelValuePair(value, count)
        chart.renderPie()

        self.dataset["charts"].append(chart.getFilename())

    def render_sensor_distribution(self):
```

```
        self._render_distribution("sensor_distribution.png", "Sensor
Distribution", "alert.analyzer.name", -1)

    def render_top10_attackers(self):
        chart = Chart('top10_attackers.png')
        chart.setTitle("Top 10 attackers")
        for address, count in
self.env.idmef_db.getValues(["alert.source.node.address.address/group_by",
                                                          "count(alert.messageid
)/order_desc"],
                                                         limit=10):

            chart.addLabelValuePair(address, count)
        chart.renderPie()

        self.dataset["charts"].append(chart.getFilename())

    def render_top10_classifications(self):
        chart = Chart("top10_classifications.png")
        chart.setTitle("Top 10 classifications")
        for classification, count in
self.env.idmef_db.getValues(["alert.classification.text/group_by",
                                                               "count(alert.me
ssageid)/order_desc"],
                                                              limit=10):

            chart.addLabelValuePair(classification, count)
        chart.renderPie()

        self.dataset["charts"].append(chart.getFilename())

   # def render_timeline(self):
   #     chart = Chart("timeline.png")
   #     chart.setTitle("Timeline")

   #     for hour in range(24):
   #         count = self.env.idmef_db.getValues(["count(alert.messageid)"],
   #                                             criteria="alert.create_time ==
'hour:%d'" % hour)
   #         chart.addLabelValuePair("%dh" % hour, count)
   #     chart.renderPlot()

   #     self.dataset["charts"].append(chart.getFilename())

    def render(self):
        self.dataset["charts"] = [ ]
        self.render_top10_classifications()
        self.render_top10_attackers()
        self.render_sensor_distribution()
        #self.render_timeline()
```

E) Module python pour l'envoi d'alerte par courrier électronique

Ce module n'est pas fourni avec la distribution de Prelude. J'ai pu le trouver sur Internet pour la version de Prelude 0.8, j'ai donc du effectuer quelques modifications de manière à ce qu'il soit compatible avec la version 0.9 et qu'il envoie des messages personnalisés. Ce module se base sur le fichier xml généré par Prelude d'où il retire ses informations. Pour les attaques s'appuyant sur des signatures, il indique les différentes ressources Web y faisant référence.

```
#!/usr/bin/python
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2, or (at your option)
# any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; see the file COPYING.  If not, write to
# the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.


import os
import sys
import select
import fcntl
import getopt

# it seems that there is a bug with the xml.dom module locales (see
# http://archives.mandrakelinux.com/cooker/2004-02/msg08160.php)
# we override the LC_ALL environnement variable to workaround this bug
os.environ["LC_ALL"] = "C"

from xml.dom import pulldom
from email import MIMEMultipart, MIMEText, Utils
from smtplib import SMTP, SMTPServerDisconnected

_USAGE = """
prelude-mail.py [OPTIONS]

-s, --server            address of the mail server (default: localhost)
-p, --server-port       port of the mail server (default: 25)
-f, --from              'From' field of the mail (default: $USER@localhost)
-t, --to                'To' field of the mail (default: $USER@localhost), one
or more
-c, --cc                'Cc' field of the mail (default: none), one or more
-l, --log               log filename to process (can be a normal file or a named
pipe) (default: /var/log/prelude-xml.log)
-d, --daemon            run prelude-mail as a daemon
-h, --help              print this help
-v, --version           print prelude-mail version
"""

class IDMEFLogFile(file):
    """
    Workaround for xml parser: we need a single root node to make the xml parser
happy,
```

```
    thus, we generate a fake 'list' node uppon the list of IDMEF-Message

    fcntl and select are done in order to avoid blocking read and buffering
(that could delay
    mail reporting)
    """
    def __init__(self, *args, **kwargs):
        file.__init__(self, *args, **kwargs)
        fcntl.fcntl(self, fcntl.F_SETFL, os.O_NONBLOCK)
        self._file_begin = True
        self._file_end = False

    def read(self, size):
        if self._file_begin:
            self._file_begin = False
            return "<list>"

        if self._file_end:
            return None

        select.select([ self ], [ ], [ ])
        buf = file.read(self) # drop the size argument so that we avoid file
internal buffering
        if not buf:
            self._file_end = True
            return "</list>"

        return buf



class PreludeMailContent:
    def __init__(self, log):
        self._log = log
        self._content = ""

    def _addContent(self, content):
        self._content += content

    def __setitem__(self, key, value):
        self._addContent("%s: %s\n" % (key, value))

    def _addSeparator(self):
        self._addContent("\n")

    def _setTime(self):
        time = self._log.getElementsByTagName("DetectTime") or
self._log.getElementsByTagName("CreateTime")
        self["Heure GMT"] = time[0].firstChild.data
        self._addSeparator()

    def _setClassification(self):
        classification = self._log.getElementsByTagName("Classification")
        if not classification:
            return
        name = classification[0].getElementsByTagName("name")
         if not name:
             return
        self["Classification"] = name[0].firstChild.data
         url = classification[0].getElementsByTagName("url")
         if not url:
             return
         self._content += "Sites de reference:\n"
         for contenu in url:
             data = contenu.firstChild.data
             self._content += "%s\n" % data
```

```
        self._addSeparator()

    def _setImpact(self):
        assessment = self._log.getElementsByTagName("Assessment")
        if not assessment:
            return

        impact = assessment[0].getElementsByTagName("Impact")
        if not impact:
            return

        severity = impact[0].getAttribute("severity")
        if severity:
            self["Niveau de l'Impact"] = severity

        completion = impact[0].getAttribute("completion")
        if completion:
            self["Reussite de l'Impact"] = completion

        type = impact[0].getAttribute("type")
        if type:
            self["Type d'Impact"] = type

        description = impact[0].firstChild
        if description:
            self["Description de l'Impact"] = description.data

        self._addSeparator()

    def _setSensor(self):
          analyzer = self._log.getElementsByTagName("Analyzer")
          if not analyzer:
              return
          analyzer = analyzer[0]
          analyzer = analyzer.getElementsByTagName("Analyzer")
          if not analyzer:
              return
          analyzer = analyzer[0]
          sensor = analyzer.getAttribute("model")
          self["Sonde"] = sensor
          analyzer = analyzer.getElementsByTagName("Analyzer")
          if not analyzer:
              return
          analyzer = analyzer[0]
          process = analyzer.getElementsByTagName("Process")
          if not process:
              return
          process = process[0]
          name = process.getElementsByTagName("name")
          if not name:
              return
          self["Processus"] = name[0].firstChild.data
          self._addSeparator()

    def _setAnalyzer(self):
        analyzer = self._log.getElementsByTagName("Analyzer")
        if not analyzer:
            return

        analyzer = analyzer[0]


        model = analyzer.getAttribute("model")
        if model:
            version = analyzer.getAttribute("version")
```

```
            if version:
                model += " %s" % version
            self["Analyseur"] = model

        ostype = analyzer.getAttribute("ostype")
        if ostype:
            osversion = analyzer.getAttribute("osversion")
            if osversion:
                ostype += " %s" % osversion
            self["Systeme d'exploitation de l'Analyseur"] = ostype

        hostname = self._getNodeName(analyzer)
        if hostname:
            self["Nom d'hote de l'Analyseur"] = hostname

        address = self._getNodeAddress(analyzer)
        if address:
            self["Adresse de l'Analyseur"] = address

        self._addSeparator()

    def _getNodeName(self, node):
        node = node.getElementsByTagName("Node")
        if not node:
            return None
        name = node[0].getElementsByTagName("name")
        if not name:
            return None
        return name[0].firstChild.data

    def _getNodeAddress(self, node):
        node = node.getElementsByTagName("Node")
        if not node:
            return None

        address = node[0].getElementsByTagName("Address")
        if not address:
            return None

        address = address[0].getElementsByTagName("address")
        if not address:
            return

        address = address[0].firstChild
        if not address:
            return None

        return address.data

    def _setDirection(self, direction):
        node = self._log.getElementsByTagName(direction)
        if not node:
            return

        node = node[0]
        count = 0

        hostname = self._getNodeName(node)
          direction = direction.replace("Target","Cible")
        if hostname:
            self["Nom d'hote %s" % direction] = hostname
            count += 1

        address = self._getNodeAddress(node)
        if address:
            self["Adresse %s" % direction] = address
```

```
            count += 1

        if count > 0:
            self._addSeparator()

    def _setSource(self):
        self._setDirection("Source")

    def _setTarget(self):
        self._setDirection("Target")

    def _setAdditionalData(self):
        ads = self._log.getElementsByTagName("AdditionalData")
        if not ads:
            return

        self._content += "Donnees Techniques:\n"

        for ad in ads:
            meaning = ad.getAttribute("meaning")
            data = ad.firstChild.data
            if meaning:
                data = "%s: %s" % (meaning, data)
            self._content += "\t%s\n" % data
         self._addSeparator()
         self._content += "Veuillez trouver ci-joint le fichier alert.xml\n"

    def __str__(self):
        self._setTime()
         self._setSensor()
        self._setClassification()
        self._setImpact()
        self._setAnalyzer()
        self._setSource()
        self._setTarget()
        self._setAdditionalData()

        return self._content



class PreludeMail:
    def __init__(self, log, sender, to, cc):
        self._log = log
        self._sender = sender
        self._to = to
        self._cc = cc

    def _getSubject(self):
        classification = self._log.getElementsByTagName("Classification")
        if not classification:
            return ""
        name = classification[0].getElementsByTagName("name")
        if not name:
             return ""
        return name[0].firstChild.data

    def _getElse(self):
         analyzer = self._log.getElementsByTagName("Analyzer")
         if not analyzer:
            return ""
         analyzer = analyzer[0]
         analyzer = analyzer.getElementsByTagName("Analyzer")
         if not analyzer:
            return ""
         analyzer = analyzer[0]
```

```
          sensor = analyzer.getAttribute("model")
          if not sensor:
             return ""
          analyzer = analyzer.getElementsByTagName("Analyzer")
          if not analyzer:
             return sensor
          analyzer = analyzer[0]
          process = analyzer.getElementsByTagName("Process")
          if not process:
             return sensor
          process = process[0]
          name = process.getElementsByTagName("name")
          if not name:
             return sensor
          name = name[0].firstChild.data
          if not name:
             return sensor
          sensor += "||%s" % name
          if not sensor:
             return ""
          return sensor

    def __str__(self):
        content = str(PreludeMailContent(self._log))
        mail = MIMEMultipart.MIMEMultipart()
          sujet = self._getSubject()
          sujet += " %s" % self._getElse()
        mail["Subject"] = "[ALERTE PRELUDE] %s" % sujet
        mail["To"] = ", ".join(self._to)
        mail["Cc"] = ", ".join(self._cc)
        mail["From"] = self._sender
        mail["Date"] = Utils.formatdate()

        message = MIMEText.MIMEText(content)
        mail.attach(message)

        message=  MIMEText.MIMEText(self._log.toxml())
        message.add_header("content-disposition", "attachment",
filename="alert.xml")
        mail.attach(message)

        return str(mail)



class PreludeMailer:
    def __init__(self):
        username = os.environ["USER"]

        self._daemon = False
        self._log_filename = "/var/log/prelude-xml.log"
        self._server = "localhost"
        self._server_port = 25
        self._sender = "%s@localhost" % username
        self._cc = [ ]
        to = [ ]

        opts, args = getopt.getopt(sys.argv[1:],
                                   "t:f:c:l:s:p:dhv",
                                   [ "to=", "from=", "cc=", "log=", "server=",
"server-port=", "daemon", "help", "version" ])

        for opt, arg in opts:
            if opt in ("-s", "--server"):
                self._server = arg
```

```
            elif opt in ("-p", "--server-port"):
                self._server_port = arg

            elif opt in ("-t", "--to"):
                to.append(arg)

            elif opt in ("-f", "--from"):
                self._sender = arg

            elif opt in ("-c", "--cc"):
                self._cc.append(arg)

            elif opt in ("-l", "--log"):
                self._log_filename = arg

            elif opt in ("-d", "--daemon"):
                self._daemon = True

            elif opt in ("-h", "--help"):
                print _USAGE.strip()
                sys.exit(0)

            elif opt in ("-v", "--version"):
                print "prelude-mail 0.0.1"
                sys.exit(0)

        self._to = to or [ "%s@localhost" % username ]

    def _processAlert(self, node):
        mail = PreludeMail(node, self._sender, self._to, self._cc)
        while True:
            try:
                self._smtp.sendmail(self._sender, ", ".join(self._to),
str(mail))
            except SMTPServerDisconnected:
                self._smtp.connect(self._server, self._server_port)
            else:
                break

    def _doDaemon(self):
        os.fork() and sys.exit(0)
        os.setsid()
        os.fork() and sys.exit(0)

    def run(self):
        if self._daemon:
            self._doDaemon()

        file = IDMEFLogFile(self._log_filename)
        events = pulldom.parse(file, bufsize=1024)
        self._smtp = SMTP()
        self._smtp.connect(self._server, self._server_port)

        for event, node in events:
            if event == pulldom.START_ELEMENT and node.tagName == "Alert":
                events.expandNode(node)
                self._processAlert(node)



mailer = PreludeMailer()
mailer.run()
```

Voici typiquement le type de message que l'on reçoit grâce à ce module :

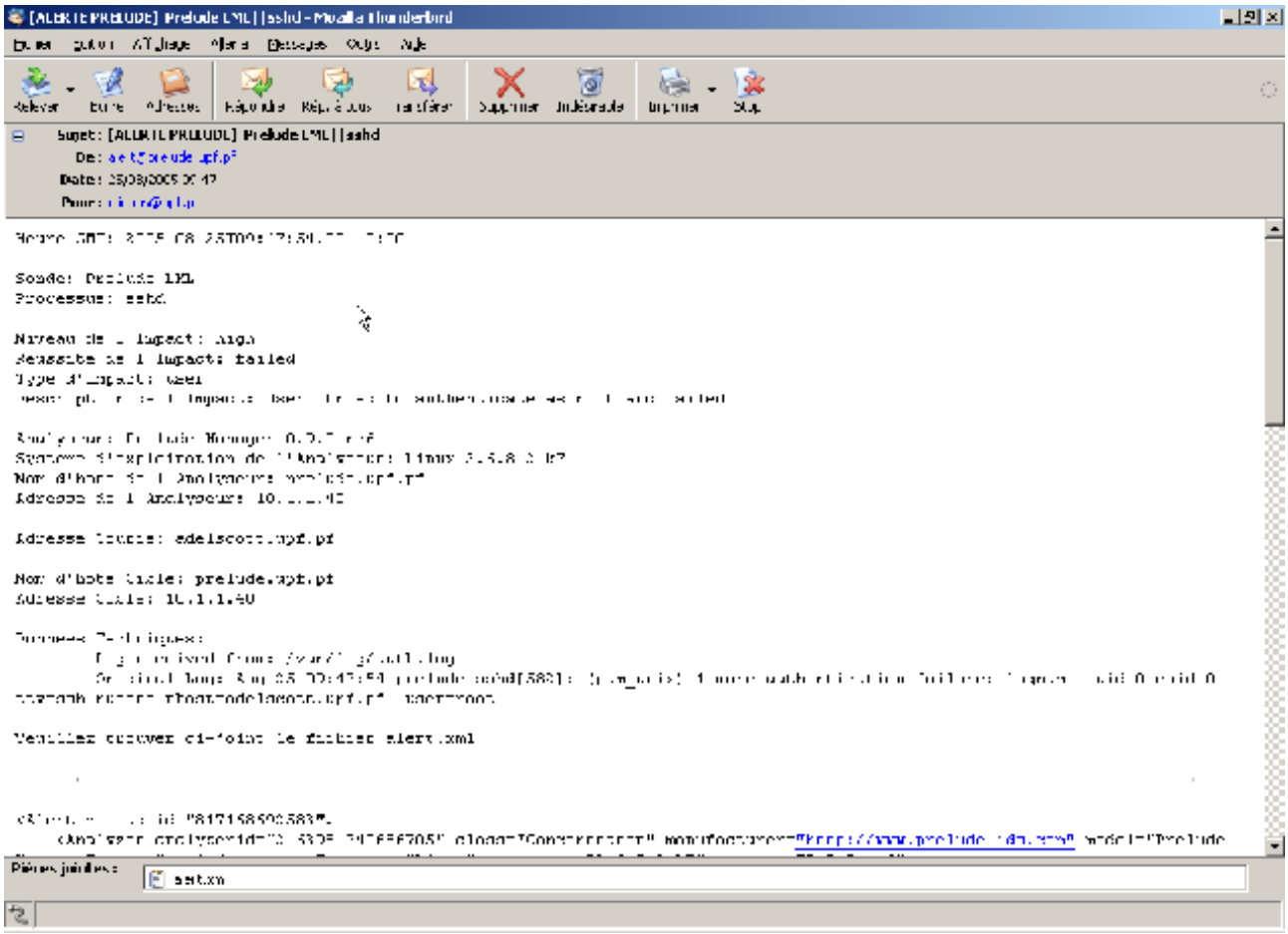


Figure 16: Message type envoyé par le module d'alertes par e-mail de Prelude

IV. Samhain

A) Compilation et installation

Vérifiez la signature PGP du fichier tar avec le fichier nommé samhain-N.N.N.tar.gz.asc puis décompressez le fichier, configurez les sources et, enfin, lancez l'installation.

```
sh$ wget http://la-samhna.de/samhain/samhain-current.tar.gz
sh$ gpg --verify samhain-N.N.N.tar.gz.asc samhain-1.8.8.tar.gz
sh$ gunzip samhain-current.tar.gz | tar tvf-
-rw-r - r - 500/100 920753 2004-05-24 19:57:55 samhain-1.8.8.tar.gz
-rw-r - r - 500/100 189 2004-05-24 19:58:29 samhain-1.8.8.tar.gz.asc
sh$ cd samhain-N.N.N
```

1. Configuration des sources

Une liste (non exhaustive) des différentes options :
Utilisation d'une base de données relationnelle : --enable-xml-log, --with-database=mysql|postgresql|oracle|odbc]
Surveillance des événements de logs : --enable-login-watch)
Vérifier le disque à la recherche de fichiers SUID/SGID suspects : --enable-suidcheck
Contrôler le noyau à la recherche de rootkits (sur Linux/FreeBSD) : --with-kcheck=SYSTEM_MAP.

2. Compilation

Après la configuration des sources, pour la compilation vous devez juste taper l'éternelle commande :

```
Sh$ make
```

Il est possible de créer nativement des packets (rpm, debian...) en fonction de la configuration des sources éffectuée précedemment.
Il suffit de taper :

```
Sh$ make rpm|deb|tbz2|solaris-pkg
```

On peut aussi construire tout simplement un paquet binaire pour Unix avec la commande :

```
Sh$ make run
```

3. Installation

Si la compilation s'est effectuée sans problème, on peut alors installer samhain en tapant :
Sh$ make install
La routine d'installation n'écrasera pas le fichier de configuration d'une installation précédente. Après l'installation, on peut lancer la copie des scripts de lancement automatique à l'amorcage de la machine, pour cela :
Sh$ make install-boot

B) Initialiser la base de données de référence

Pour exécuter l'initialisation (c'est-à-dire créer la base de données de référence), il faut taper :
Sh$ samhain -t init -p info
Si le fichier de base de données de référence existe déjà, samhain -t init rajoutera les références à la fin du fichier.
Pour mettre à jour cette base, tapez samhain -t update.

C) Exécuter Samhain

sh$ samhain -t check
Pour exécuter samhain comme un démon, vous pouvez utiliser l'option de ligne de commande '-d ' ou installer le mode démon dans le fichier de configuration (/etc/samhainrc) avec l'option 'Daemon=yes'.
Vous pourrez ainsi utiliser directement le script fourni :
sh$ /etc/init.d/samhain start
La base de données de référence se trouve dans /var/lib/samhain/samhain_file, le fichier pid dans /var/run/samhain.pid et le fichier de journal dans /var/log/samhain_log.
Le serveur Yule écrit en plus un fichier de statut des clients HTML dans /var/log/yule/yule.html
Pour l'utilisation de samhain en mode client/serveur avec la base de données de référence se situant sur le serveur, il y a deux façons de mettre à jour la base de données :
la meilleure méthode se base sur l'interface Web beltane.
Cette interface permet de passer en revue les messages des clients et d'exécuter les mises à jour côté serveur des bases de données de référence.

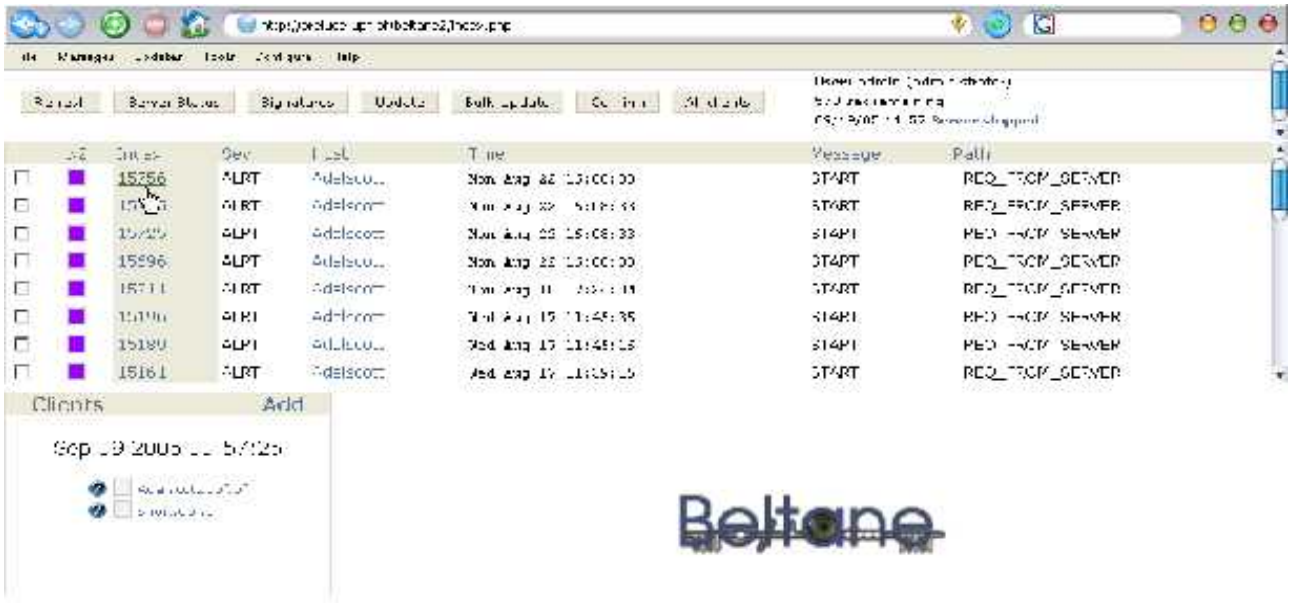


Figure 17: Interface Web Beltane pour la gestion client/serveur de Samhain

On peut aussi temporairement faire une copie distante de la base de données
Pour cela, exécutez samhain -t update et copier la base de données sur le serveur.

D) Envoi automatique de courriers électroniques

Les paramètres à configurer dans le fichier de configuration sont :
Adresse du Destinataire
SetMailAddress=username@hostname
Jusqu'à huit adresses sont possible, chacune de 63 caractères de longueur, chacune sur une ligne séparée.On recommande d'utiliser des adresses IP numériques au lieu des noms d'hôte (pour éviter l'usage du serveur DNS).
Hôte de relais
SetMailRelay=mail.some_domain.com
Vous pouvez avoir besoin de cette option parce que quelques sites ne permettent pas les connexions du courrier électronique en partance de n'importe quel hôte arbitraire.
Intervalle maximal
SetMailTime=86400
Vous pouvez vouloir mettre un intervalle maximal entre deux courriers consécutifs

afin d'être sûr que samhain est toujours 'vivant'.
Nombre maximal de mails en suspens
SetMailNum=10
L'envoi de messages respecte un seuil d'alerte. Ainsi ceux avec une haute sévérité seront placés en premier dans la file d'attente. Au plus 128 messages peuvent être mis en file d'attente.
Destinataires multiples
MailSingle=yes/no
Si tous les destinataires sont sur le même domaine, un courrier simple suffit.
Ligne "objet"
MailSubject=string
Ici, la chaîne peut contenir les espaces réservés %T, %H, %S, et-ou %M qui seront remplacé par l'heure GMT, le nom d'hôte, la sévérité du message et enfin le contenu du message. La ligne "objet" par défaut est '%T %H".
Expéditeur
SetMailSender=string
Si on donne un nom sans domaine (C'est-à-dire sans '@upf.pf'), le FQDN (nom d'hôte + domaine) local sera ajouté automatiquement.

E) Compatibilité avec Prelude 0.9

Pour libprelude 0.9, le détecteur par défaut se nomme 'samhain'.
il existe une option PreludeProfile=profile (dans la section [Misc]) afin de définir un autre profil par défaut.
La procédure est la même pour l'ensemble des modules reliés à Prelude.Il faut lancer la commande prelude-adduser sur le client ainsi que sur le manager.
Vous devez taper le mot de passe unique produit par "le manager".
Alors "le manager" vous demandera d'approuver l'enregistrement.

F) Utilisation de samhain avec nagios

Il existe aussi un script appelé check_samhain.pl pour l'intégration de samhain dans nagios.
Dans votre fichier /etc/sudoers, ajoutez la ligne :
Nagios ALL = NOPASSWD:/path/to/check_samhain
Ensuite, ajoutez le plugin dans le fichier checkcommands.cfg de nagios:
# 'check_samhain' command definition
define command{
command_name check_samhain
command_line /usr/bin/sudo u root $USER1$/check_samhain t 100
}
La vérification du système de fichiers peut prendre du temps.

Il vaut mieux augmenter le timeout des services (de 60 à 100) dans nagios.cfg :
Service_check_timeout=100

Détection de rootkits
Cette option est actuellement supportée pour Linux avec noyau 2.2.x, 2.4.x et 2.6.x sur machines ix86 et sur FreeBSD (testé sur FreeBSD 4.6.2).
Pour utiliser cette fonction, il faut avoir compiler samhain avec l'option :
./configure --with-kcheck=/path/to/System.map (Linux)
./configure --with-kcheck (FreeBSD).
Sur Linux, System.map est un fichier (parfois avec la version du noyau ajoutée à son nom) qui est produit quand le noyau est compilé et est en général installé dans le même répertoire que votre noyau. (Par exemple /boot) ou dans le répertoire racine.

V. Scripts de démarrage

Aucun script de démarrage n'est fourni avec la majorité des outils énoncés dans ce rapport, je les ai donc crée. Il en existe un pour Nessus, un pour Prelude-lml, un pour Prelude-manager, un pour Prewikka et un pour la remontée d'alertes par e-mail.

A) Nessusd

```
#! /bin/sh

# Nessus Daemon start/stop script.


PATH=/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin
DAEMON=/sbin/nessusd
PID=/var/nessus/nessusd.pid
OPTION='-D'
NAME=nessusd
DESC="Nessus Daemon"

test -f $DAEMON || exit 1

set -e

case "$1" in
  start)
          echo "Starting $DESC: $NAME"
          start-stop-daemon --oknodo --start \
                    --exec $DAEMON -- $OPTION
          ;;
  stop)
          echo "Stopping $DESC: $NAME"
          start-stop-daemon --oknodo --stop  \
                    --pidfile $PID --name $NAME
          ;;
  status)
          if [ -s $PID ]; then
          pid_num=`cat $PID`
          kill -0 $pid_num >/dev/null 2>&1
          if [ "$?" = "0" ]; then
          echo "Nessus server (pid $pid_num) is running"
          else
          echo "Nessus server is stopped"
          fi
          else
          echo "Nessus server is stopped"
          fi
          ;;
  restart|force-reload)
          $0 stop
          $0 start
          ;;
  *)
          N=/etc/init.d/$NAME
          # echo "Usage: $N {start|stop|restart|force-reload}" >&2
          echo "Usage: $N {start|stop|restart|force-reload|status}" >&2
          exit 1
          ;;
esac

exit 0
```

B) Prelude-manager

```
#! /bin/sh
#
# Prelude manager start/stop script.
#

PATH=/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin
DAEMON=/usr/local/bin/prelude-manager
PID=/usr/local/var/prelude/prelude-manager.pid
OPTION='-d -P'
NAME=prelude-manager
DESC="Prelude Manager"

test -f $DAEMON || exit 0

set -e

case "$1" in
  start)
          echo "Starting $DESC: $NAME"
          start-stop-daemon --oknodo --start \
                    --exec $DAEMON -- $OPTION $PID
          ;;
  stop)
          echo "Stopping $DESC: $NAME"
          start-stop-daemon --oknodo --stop \
                    --pidfile $PID --name $NAME -- $OPTION $PID
          ;;
  status)
          if [ -s $PID ]; then
        pid_num=`cat $PID`
        kill -0 $pid_num >/dev/null 2>&1
        if [ "$?" = "0" ]; then
        echo "$DESC (pid $pid_num) is running"
        else
        echo "$DESC is stopped"
        fi
        else
        echo "$DESC is stopped"
        fi
        ;;
  restart|force-reload)
          $0 stop
          $0 start
          ;;
  *)
          N=/etc/init.d/$NAME
          echo "Usage: $N {start|stop|restart|force-reload}" >&2
          exit 1
          ;;
esac

exit 0
```

C) Prelude-lml

```
#! /bin/sh
#
# Prelude lml start/stop script.
```

```
#

PATH=/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin
DAEMON=/usr/local/bin/prelude-lml
PID=/usr/local/var/prelude/prelude-lml.pid
OPTION='-d -P'
NAME=prelude-lml
DESC="Prelude lml"

test -f $DAEMON || exit 0

set -e

case "$1" in
  start)
        echo "Starting $DESC: $NAME"
        start-stop-daemon --oknodo --start \
                --exec $DAEMON -- $OPTION $PID
        ;;
  stop)
        echo "Stopping $DESC: $NAME"
        start-stop-daemon --oknodo --stop \
                --pidfile $PID --name $NAME
        ;;
  status)
         if [ -s $PID ]; then
       pid_num=`cat $PID`
       kill -0 $pid_num >/dev/null 2>&1
       if [ "$?" = "0" ]; then
       echo "$DESC (pid $pid_num) is running"
       else
       echo "$DESC is stopped"
       fi
       else
       echo "$DESC is stopped"
       fi
       ;;
  restart|force-reload)
        ;;
  *)
        N=/etc/init.d/$NAME
        # echo "Usage: $N {start|stop|restart|force-reload}" >&2
        echo "Usage: $N {start|stop|restart|force-reload|status}" >&2
        exit 1
        ;;
esac

exit 0
```

D) Prelude-mail

```
#! /bin/sh
#
#

PATH=/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin
DAEMON=/usr/bin/prelude-mail
NAME=prelude-mail
DESC=prelude-mail
OPTION='-s mailhost.upf.pf -f alert@prelude.upf.pf -t cri-systeme@upf.pf -d'

test -f $DAEMON || exit 0

set -e
```

```
case "$1" in
  start)
        echo "Starting $DESC: $NAME"
        start-stop-daemon --oknodo --start \
                --exec $DAEMON -- $OPTION
        ;;
  stop)
        echo "Stopping $DESC: $NAME"
        start-stop-daemon --oknodo --stop \
                --name $NAME
        ;;
  restart|force-reload)
        $0 stop
        $0 start
        ;;
  *)
        N=/etc/init.d/$NAME
        # echo "Usage: $N {start|stop|restart|force-reload}" >&2
        echo "Usage: $N {start|stop|restart|force-reload}" >&2
        exit 1
        ;;
esac

exit 0
```