Infrastructure WIFI, mise en place et sécurisation

Rapport de stage de 2ème année informatique

Mathieu GELI

Juin-Août 2005

Table des matières

Table des figures

Remerciements

Je tiens tout d'abord à remercier mon maître de stage Laurent Facq qui m'a accompagné tout au long de ce stage. Sa disponibilité et sa passion m'ont été riche d'enseignements. Je remercie également Grégoire Moreau et Ludovic Doit dont la collaboration fut fructueuse et qui ont toujours pu me donner des avis et des conseils enrichissants, et bien entendu, tout le personnel de Réaumur.

Introduction

Dans le cadre du projet **ACO** [1], l'appropriation de l'espace numérique par les étudiants est mise en avant. Ceci passe donc par une installation sans-fil de type Wifi, afin de mettre à disposition des étudiants un accès en ligne dans le périmètre du campus.

L'utilisation d'un médium non physique entraîne des problèmes de sécurité auxquels ils faut faire face de manière à minimiser les risques. Les accès Wifi se démocratisant en parallèle des accès ADSL haut-débit, la sensibilisation sur l'aspect sécurité n'est pas toujours à la hauteur. On trouvera souvent en agglomération des accès de particulier non ou faiblement sécurisés. C'est donc une aubaine pour la personne nomade mal intentionnée voulant masquer son identité.

Le service **Réaumur** [2], a été chargé de la réalisation technique de l'infrastructure logicielle. C'est donc dans ce cadre de mise en oeuvre et en rajoutant cette problématique de sécurité que s'inscrit l'objectif de mon stage.

[1] Aquitaine Campus Ouvert

[2] Réseau Aquitain des Utilisateurs en Milieu Universitaire et de Recherche

Chapitre 1

Le service Réaumur

1.1 Origine

Le réseau Renater [1] existe depuis 1991. Son but est de inter-connecter les différentes universités françaises entre elles ainsi qu'à Internet.

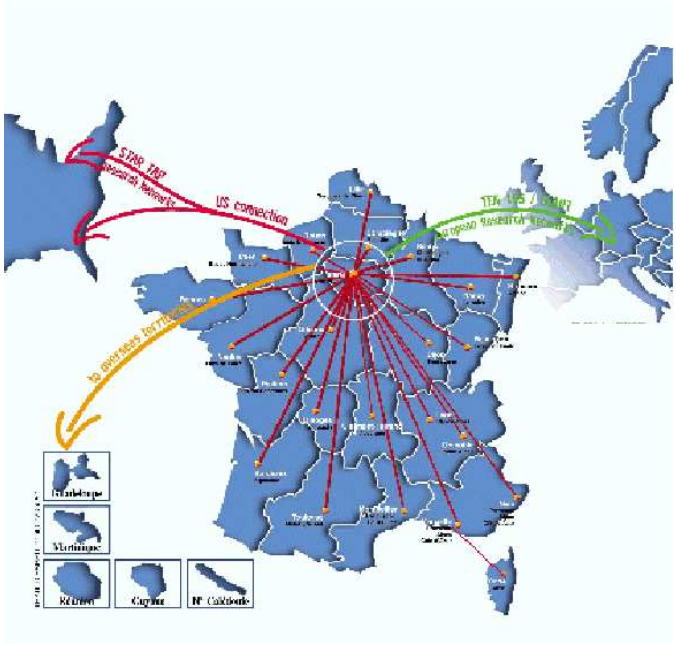

Fig. 1.1 – Topologie du réseau Renater

[1] Réseau National de télécommunications pour la Technologie l'Enseignement et la Recherche

En aquitaine le besoin de créer un service régional d'interconnexion des différents organismes universitaires s'est fait ressentir ce qui a donné naissance au service Réaumur.

1.2 Missions

Le service Réaumur est un service inter-universitaire qui a été créé afin de prendre en charge le réseau informatique de campus à haut débit.

Une de ses mission pour les partenaires du campus Talence, Pessac, Gradignan est d'assurer l'exploitation, la gestion et le développement de moyens et de services réseaux communs, dans le cadre des activités scientifiques, pédagogiques, documentaires ou de gestion des différents centres, laboratoires et services des organismes partenaires. Réaumur fixe notamment les conditions de sécurité d'utilisation, en accord avec les partenaires.

Le service de base réalisé par Réaumur consiste à mutualiser les services de connexion à Renater.

Réaumur a également une mission élargie au plan régional, puisqu'il est gestionnaire de la Plaque réseau régionale ESRA pour les établissements d'Enseignement Supérieur et de Recherche en Aquitaine. Cette mission concerne les partenaires déjà cités auxquels s'ajoutent le Rectorat de Bordeaux, le CEMAGREF, l'EAPBx, l'ENITA de Bordeaux, l'INRA Aquitaine, l'IUFM d'Aquitaine, l'Université de Pau et des Pays de l'Adour, et enfin l'Ecole de Management de Bordeaux. Ses missions sont coordonnées par la DRIMM. [2]

1.3 Conseil d'administration

Réaumur est gérée par un Conseil d'Administration et dirigée par un Directeur.

Il comprend, au 01/01/2004, 17 membres de droit :

- les Présidents, Directeurs, Délégués des 10 organismes signataires (voir annexe 1), ou leurs représentants,
- le Vice-Président chargé de la recherche de l'Université Bordeaux 1 ou son représentant,
- le Directeur de l'IUT Bordeaux 1 ou son représentant,
- le Directeur de l'IUT Michel de Montaigne ou son représentant,
- le Directeur de l'IUT Bordeaux Montesquieu ou son représentant,
- le Directeur de Réaumur ,
- le responsable du pôle plaque urbaine,

[2] Direction des Ressources Informatiques et Multimédia Mutualisées

– le responsable du pôle réseau régional,

15 membres élus ou désignés :

– 2 représentants du personnel technique de Réaumur ,
– 11 représentants choisis par le Groupe des Utilisateurs,
– 2 personnalités extérieures (une personnalité proposée par le Conseil Régional d'Aquitaine, une personnalité proposée par la Communauté Urbaine de Bordeaux).

Le conseil d'administration se réunit au moins une fois par an, son rôle principal est de définir la politique générale de développement du service, définir les conditions d'utilisation des moyens au niveau de la sécurité et détermine la contribution financière des organismes signataires.

1.4 Personnel

L'équipe Réaumur est composée de 7 personnes :

– 2 ingénieurs de recherche : directeur (gestion de l'administratif et des ressources humaines) et directeur technique (chef de projet, développement, sécurité du réseau).
– 1 ingénieur d'étude (développement et gestion quotidienne du réseau),
– 1 assistant ingénieur (assistance aux utilisateurs),
– 1 technicien (gestion de la partie physique du réseau),
– 1 secrétaire (assistance au directeur, gestion du site Web du service),
– 1 apprenti (assistance aux ingénieurs, développement).

1.5 Dispositions financières

Les ressources de Réaumur sont constituées en ce qui concerne la couverture des dépenses liées à l'exploitation directe du réseau Réaumur , par les contributions des organismes signataires, calculées conformément aux modalités définies annuellement par le Conseil d'Administration. Ces dépenses incluent la connexion à la plaque urbaine et au réseau régional.

Réaumur peut recevoir des subventions d'organismes publics ou privés et conclure des contrats. Elle peut enfin facturer des prestations exceptionnelles.

Les dépenses et recettes de Réaumur sont inscrites dans un compte séparé de l'Université de rattachement. Elles sont ordonnancées par le Président de l'Université et réglées par son Agent Comptable.

1.6 Concurrence

Les concurrents potentiels de Réaumur sont les sociétés privées de gestion de réseau qui pourraient être choisies pour remplacer toute ou partie de l'équipe titulaire.

A ce sujet, la gestion du réseau ESRA (noeud de raccordement de la plaque universitaire bordelaise) était confiée à des sociétés privées, jusqu'à ce qu'en 2004 Réaumur le gère complètement.

1.7 Topologie du réseau

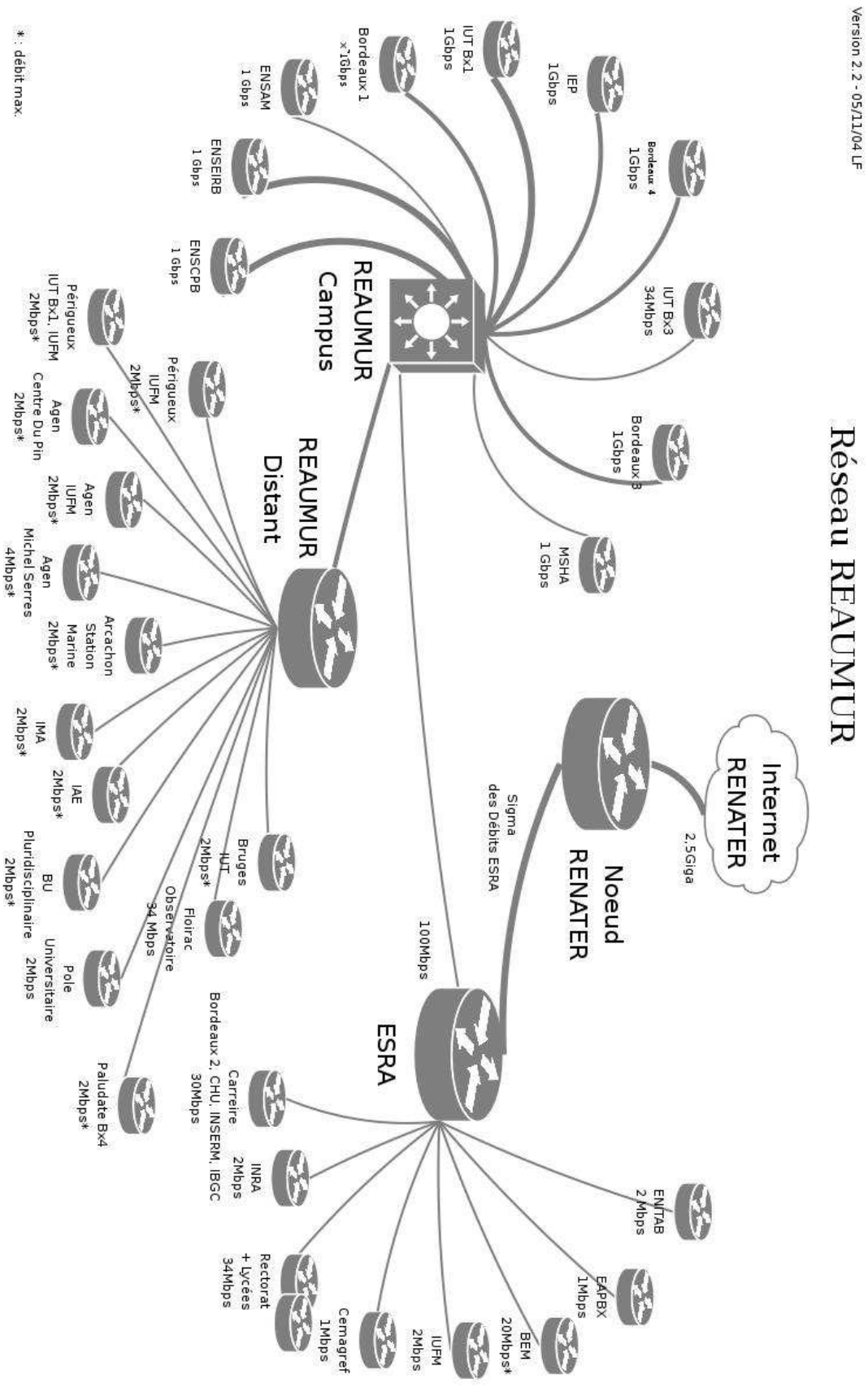

FIG. 1.2 – Topologie du réseau Réaumur

Chapitre 2

Objectifs de la mission technique

2.1 Sujet de stage

"Mise en place de portail captif Wifi sur réseau Réaumur et affiliés. Étude de la mise en oeuvre de protocoles plus sécurisés ainsi que l'intégration du dispositif dans des projets d'authentification à plus grande échelle"

2.2 Cahier des charges fonctionnel

Le service Réaumur doit mettre en place une infrastructure wifi opérationnelle en septembre 2005. Cette infrastructure doit être accessible à l'ensemble des étudiants et personnels des partenaires de l'UNR[1] ACO[2] et leur permettra d'accéder au web. La partie physique (mise en place des bornes) est assurée par une entreprise externe et sera terminée courant Juin. Il reste à mettre en place la partie contrôle d'accès, sécurisation et supervision.

Dans un premier temps, afin d'avoir un système opérationnel rapidement et de permettre à toutes les personnes autorisées d'y accéder, le contrôle d'accès se fera par portail captif.

Dans un deuxième temps, sera étudié la mise en oeuvre de protocoles plus sécurisés (802.1x , 802.11i , serveur radius) ainsi que l'intégration du dispositif dans des projets de authentification à plus grande échelle (projets du CRU [3], Arredu [4] et fédération d'identités).

[1] Université Numérique Régionale
[2] Aquitaine Campus ouvert
[3] Comité Réseau des Universités
[4] Authentification Répartie Recherche ÉDUcation

Enfin, ayant certains partenaires à distance on voudrais étudier une solution de tunneling de manière à raccorder les WLAN distant au portail.

Le stagiaire prendra à sa charge une part de ces travaux au sein de l'équipe en charge du projet. L'ensemble sera réalisé en utilisant au maximum des logiciels libres qu'il conviendra d'étudier, d'adapter et d'améliorer.

2.3 Contexte

2.3.1 Le projet ACO

Le projet ACO regroupe les quatre universités bordelaises, l'Université de Pau et des Pays de l'Adour et l'IUFM d'Aquitaine au sein d'un projet global ambitieux de la modernisation de la quasi-totalité des systèmes d'information universitaires en Aquitaine.

Plus précisément il s'axe autour des objectifs suivants :

- Faciliter l'accès à distance aux ressources numériques, l'équipement individuel et collectif, les usages collaboratifs Valoriser les établissements universitaires aquitains en ouvrant largement l'accès public aux savoir et aux résultats de leur activité pédagogique et scientifique.
- Moderniser les équipements informatiques et l'infrastructure des réseaux de communication à haut débit
- Homogénéiser, rendre inter-opérable, dynamiser les systèmes d'information et de communication

C'est donc dans ce cadre que s'inscrit la volonté de permettre un accès à l'espace numérique pour les étudiants, ce qui techniquement nécessite une infrastructure réseau Wifi.

La situation de Réaumur est donc celle d'un prestataire de service, qui mets à disposition les ressources réseau pour permettre l'interconnection des différents réseau Wifi du campus. Une composante importante est que le service Réaumur ne veut pas voire transiter sur son réseau des informations d'authentification comme des mots de passe des usagers en clair.

2.3.2 Arredu

Arredu est un projet d'architecture d'authentification répartie, utilisant le protocole Radius, entre les établissements (au sens de Renater) d'enseignement supérieurs et de recherche français. Cette authentification vise à offrir des accès réseau sans fil aux membres de la communauté concernée en déplacement sur les sites des partenaires avec leurs nom et mot de passe

habituels. Cette architecture devra pouvoir s'intégrer dans le projet européen EduRoam et sa future déclinaison EduRoam-ng.

Le projet est piloté par la cellule technique du CRU et regroupe, dans un premier temps, un petit nombre de participants ayant déjà une infrastructure d'authentification Radius et prêts à contribuer à la définition et la mise en place d'un pilote.

2.4 Critères de validation du projet

La validation des différentes étapes du projet est basé sur plusieurs critères. Tout d'abord dans le choix des différentes solutions logicielles adoptées. Ce choix a bien sûr été accompagné par le maître de stage. Ensuite chaque partie du projet a pu être testé en simulant des conditions d'utilisation réelles, on directement en conditions réelles. Le comportement attendu validant le test. Une partie important du stage ayant attrait à l'administration système, la validation d'un logiciel se résume en un fonctionnement en adéquation avec les modifications apportées dans sa configuration.

De plus, la mise en production de l'infrastructure n'étant pas à la charge du stagiaire, le validation progressive de la maquette permettra à la transition de se dérouler en minimisant les surprises.

2.5 Moyens mis à disposition

2.5.1 Matériel

Pour faire la maquette de test est mis à disposition :

- un serveur sous Debian GNU/Linux qui fera tourner les services principaux nécessaire au projet. Il est équipé de deux cartes réseaux, une pour le portail captif, et l'autre pour l'accès au net.
- un laptop personnel sous Debian GNU/Linux pour les tâches de développement et de test comme client wifi.
- un switch Cisco Catalyst 2950
- une borne Cisco Aironet 1130 B/G

2.5.2 Ressources réseau

Un VLAN "stagiaire" a été mis à disposition. Celui ci est routé pour pouvoir accéder à internet. Le serveur de test dispose d'une patte avec adresse IP publique *147.210.6.253*. Le serveur de nom central a été mis à jour pour répondre à cette adresse IP par le nom de *nocatauth.u-bordeaux.fr*

Ceci constitue les modifications préalable qui ont du être réalisée sur les équipements existant en production afin de mettre en place la plateforme de test. Toutes les modifications futures sur l'infrastructure étaient prévues pour ne pas impacter les systèmes en production.

2.6 Planification du projet

Voici un diagramme de Gantt permettant de visualiser l'enchaînement et la durée des différentes tâches durant le projet.

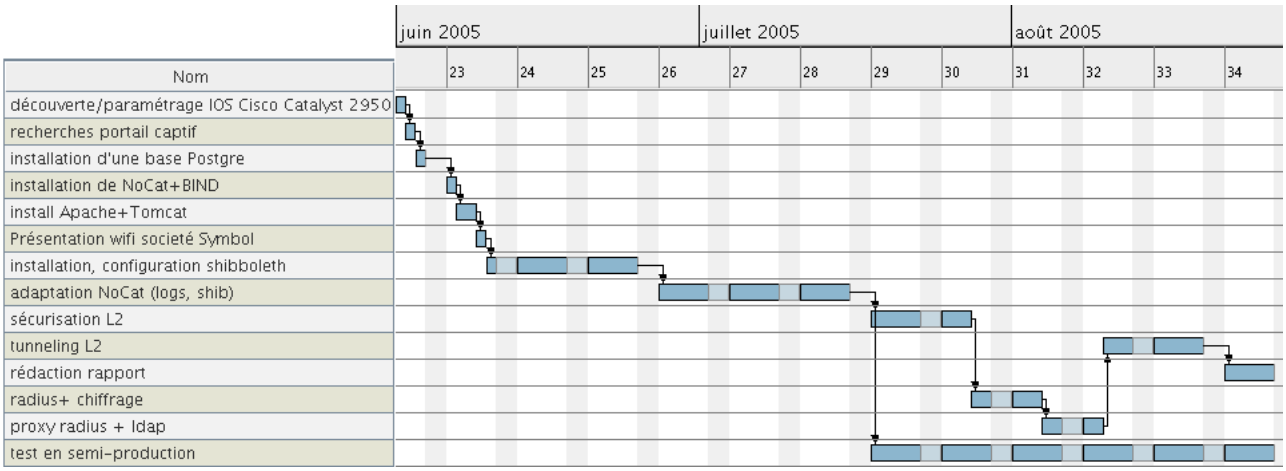

Fig. 2.1 – Planification des tâches

Le diagramme représente des tâches pour la plupart non recouvrantes, mais dans la pratique, il a bien entendu été à plusieurs reprises question de revenir sur des parties censées être achevées pour corriger ou modifier l'infrastructure.

2.7 Déroulement du projet

La première partie du projet où l'on voit plusieurs petites tâches est la mise en place de l'infrastructure de test, ce qui comprends la prise en main du matériel mis à disposition ainsi que l'installation logicielle des différents services essentiels pour la poursuite des évènements.

La deuxième phase importante a été l'installation et la configuration de la brique logicielle Shibboleth jouant le rôle dans l'infrastructure actuelle permettant de véhiculer une authentification de manière sûre pour l'accès à

une ressource à partir d'une zone partenaire. Le fonctionnement de shibboleth sera détaillé plus loin.

Après avoir avec succès installé les différents composants shibboleth, et ayant une base de services fonctionnant, la 3ème partie c'est axée sur l'installation et l'adaptation aux besoin de Réaumur du portail captif NoCat.

Ayant une infrastructure fonctionnelle remplissant la première partie du cahier des charges, on a pu commencer à faire des tests toujours avec les outils de la maquette et en rajoutant les bornes Wifi ACO. Les étudiants de passages dans les différents sites ont donc permis d'observer les comportements indésirables, et de créer des scénarios qui sont difficilement simulable sur la maquette. La phase de test avec utilisateurs a duré jusqu'à la fin du stage.

Après le travail sur NoCat, une étude de sécurisation des équipements principalement au niveau 2 (commutateur, bornes) a été faite.

Ensuite afin d'accroître la sécurité de l'accès au lien wifi, et au chiffrement des données y transitant, ont été étudiées des solutions à base de serveur radius. Les tests ont été effectués avec différents types d'authentification, souvent imposés par le support côté client des-dits protocoles.

En dernière étape a été étudié une solution pour pouvoir joindre deux LANs au niveau 2 sur un lien IP. Le cas de figure pouvant se poser si l'on veut gérer avec un unique portail captif l'accès à l'internet de partenaires connectés au réseau Réaumur par des liaisons opérateurs.

Chapitre 3

Réalisation du projet

3.1 Analyse du cahier des charges et de ses contraintes

Accès au web

On veut que les étudiants puisse accéder uniquement à du contenu Web. Il est donc nécessaire d'effectuer un filtrage au niveau de la couche de transport des ports contactés. Parallèlement, on veut que toute demande d'accès au web passe par le système d'authentification du portail, d'ou la dénomination "captif".

Authentification

L'infrastructure wifi est répartie sur tout le campus, on baigne donc dans un milieu pluri-universitaire. Ces personnes doivent pouvoir s'authentifier auprès de leur université d'origine dans un canal crypté de bout en bout.

Restriction temporelle

On ne veux pas de connections permanente sans vérification régulière de l'identité de celui qui est connecté, pour des raisons évidentes de sécurité. Il faut donc avoir un système qui scrute chaque connection et demande une re-authentification si le temps imparti est écoulé.

Journalisation

On va devoir garder pendant un certains temps les informations de connection suivantes :

- qui
- quand
- d'où
- quantité de données reçue/émise
- destinataire de la connection
- protocoles utilisés Ceci afin de pouvoir mieux diagnostiquer les problèmes liés aux abus, ou simplement les problèmes techniques.

Solution 802.11i

La deuxième partie après le système de portail captif nécessite pour les utilisateurs de Windows d'avoir un système récent, sachant que sous Linux le problème est résolu au cas par cas. De plus la solution de chiffrement utilisant AES[1], très gourmande en ressources nécessite une puce dédiée ce qui ne permet donc pas une simple mise à jour pour profiter de la norme. Néanmoins la première version de 802.11i ne standardise pas l'utilisation d'AES, et permet d'avoir de meilleurs résultats en terme de sécurité que WEP.

Cette solution utilise le standard 802.1x pour le contrôle d'accès au réseau. Il faut donc mettre en place un serveur radius qui fera l'authentification. Ca implique d'avoir des bornes Wifi compatibles pouvant discuter avec un radius. De plus pour avoir la possibilité de s'intégrer plus tard dans un réseau de type Arredu, il faut pouvoir proxifier les requêtes radius de manière à offrire la possibilité à une personne vacante de pouvoir s'authentifier sur le radius de son université/laboratoire d'origine.

Tunneling ethernet

Pour ce dernier élément afin de réaliser une solution de tunneling pour chaque site distant une étude est nécessaire sur le type de machine à mettre en place sur les sites partenaire. Il va également falloir se mettre d'accord sur une solution logicielle.

3.2 Conception générale

3.2.1 Topologie de l'environnement portail captif

Dans cette configuration de portail captif, nous avons une borne qui nnonce un certain réseau. A ce réseau on associe un VLAN, côté interface

[1] Advanced Encryption Standard

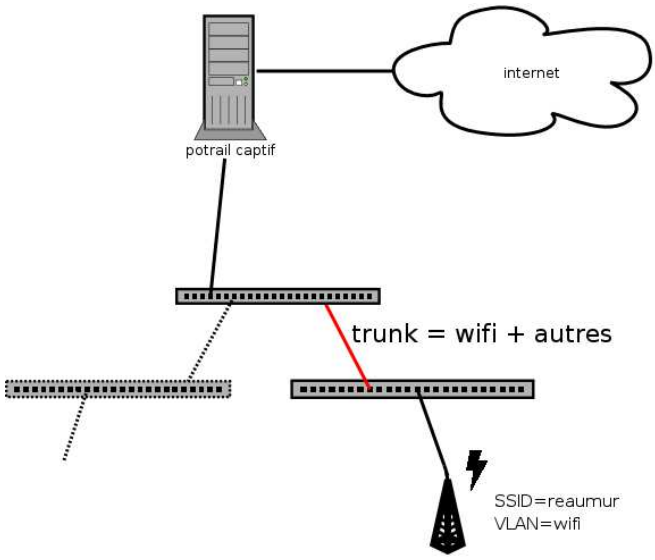

FIG. 3.1 – Topologie réseau avec portail

ethernet de la borne. Il est également connu dans le commutateur sur lequel est connecté la borne, ainsi que dans tout les commutateurs qui relient le serveur portail et la borne. Cette mesure est faite pour cloisonner le réseau des utilisateurs du wifi par rapport à l'infrastructure existante de Réaumur.

3.2.2 Portail captif

Dans le milieu open source il existe plusieurs solutions de portail captif, celle qui a retenu notre attention est nommée NocatAuth. Cette implémentation est écrite en Perl, langage amplement utilisé dans le service, fonctionne sous GNU/Linux avec comme couche basse le pare-feu iptables. Il est de plus fourni avec plusieurs moyens d'authentification. C'est un portail qui peut être modifiable est debuggable facilement. C'est donc un candidat intéressant.

Le client qui cherche à accéder à un site, va être redirigé sur la partie 'gateway' de NoCat par le biais de règles de redirection iptables. Cette gateway est un service Perl à l'écoute sur un port particulier. Ce service génère alors un jeton, et demande à la partie authentification nocat de prendre le relais tout en lui fournissant les informations dont il dispose.

Initialement le client ne peut pas sortir de son sous réseau, car la passerelle nocat bloque l'accès vers l'extérieur. Dès qu'il y a confirmation de la part de la partie authentification la passerelle rajoute des règles dans le firewall pour relâcher l'accès au client.

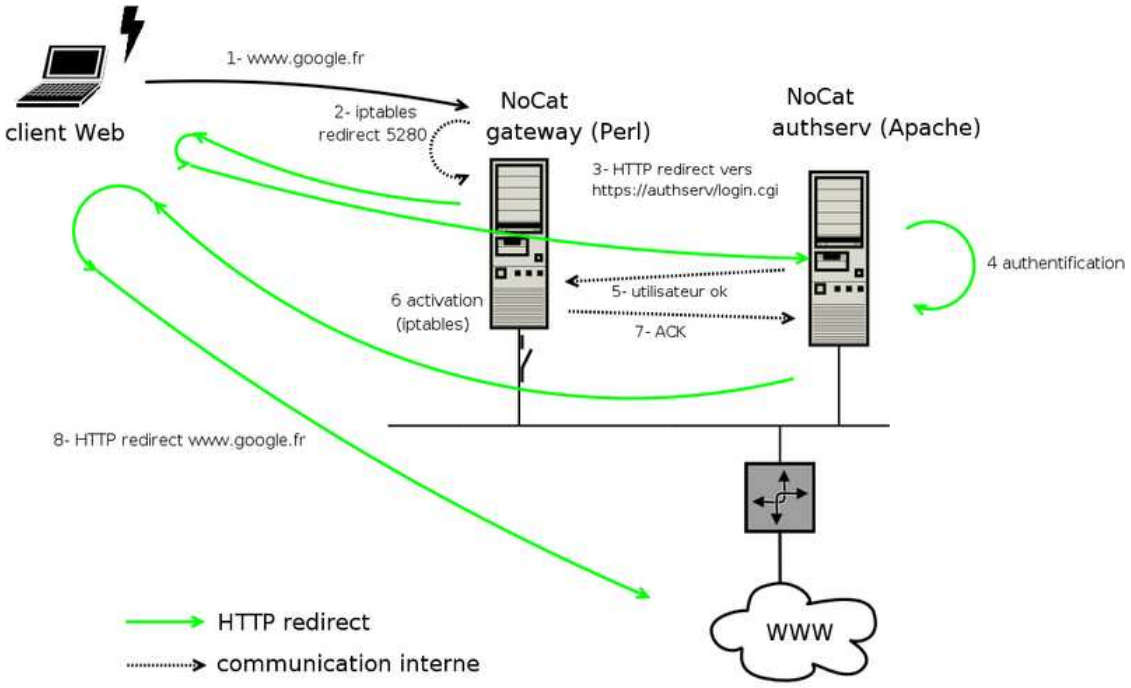

FIG. 3.2 – Flux d'information dans NoCat

Nocat est livré tel quel avec plusieurs possibilités d'interfacage d'un système authentifiant, cependant ils ne sont pas satisfaisant quand on considère le cadre dans lequel le logiciel sera employé. On ne veux pas avoir à gérer l'authentification de ces utilisateurs d'origine diverse. Il faut donc intégrer une solution d'authentification qui réponde mieux à nos besoins en adaptant le code de nocat. On s'est tourné vers la solutions logicielle shibboleth qui permet de mettre en place de l'authentification inter-établissement par le biais d'une fédération d'identité.

3.2.3 Authentification et contrôle d'accès au portail

Il existe déjà dans quelques universités un système mis en place pour l'authentification nommée CAS[2]. Il faut un moyen de pouvoir authentifier les étudiants de manière sûre en les renvoyant sur le CAS de leur université. CAS est une implémentation du principe SSO[3]. C'est à dire qu'un fois authentifié, le client dispose d'un jeton qui lui permet de faire valoir son identité. Dès lors, n'importe quel service se basant sur ce jeton peut demander

[2]Central Authentication Service
[3]Single Sign On

confirmation de sa validité au serveur CAS et dans l'affirmative autoriser le client à accéder à ses ressources. Le jeton est stocké au niveau du client par un cookie dans le navigateur. Un système assez récent qui utilise pleinement ses fonctionnalités en allant plus loin dans la sécurisation des transfert est Shibboleth. Ces auteurs le définissent de la sorte :

> *Shibboleth is an initiative to develop an open, standards-based solution to the needs for organizations to exchange information about their users in a secure, and privacy-preserving manner*

Ceci dépasse donc le cadre de l'authentification qui est qu'une partie de l'infrastructure Shibboleth. On a décidé d'utiliser ce système pour plusieurs raisons :

- de par la sécurité qu'il offre dans les protocoles employés
- du fait que le projet soit open source et donc adaptable à moindre frais
- il réponds pleinement aux besoins énoncés de contrôle d'accces à des ressources pour des personnes de différentes origines (universités)

Détaillons un peu plus le fonctionnement du framework Shibboleth.

3.2.4 Shibboleth

Shibboleth se base sur plusieurs concepts clé :

Fédération de sites : L'université d'origine de le personne fournit certains de ses attributs aux partenaires chez qui elle essaye d'accéder à des ressources. Il y a donc un réseau formé entre les établissements permettant d'identifier l'utilisateur et lui assigner un niveau de confiance. Les sites d'origine sont responsable de l'authentification des usagers, et ont la liberté de le faire par n'importe quel moyen valable.

Contrôle d'accès par attributs : Les décisions de contrôle d'accès sont basées sur ces assertions. L'identité d'un individu peut être comprise dans ces attributs, bien que dans certains cas ce ne soit pas nécessaire. Par exemple, des support de cours en ligne accessible à uniquement quelques étudiants d'un module spécifique.

Gestion actives des informations privées : Une personne peut contrôler sous certaines conditions quels seront les attributs qui seront divulgués par son site d'origine à partir de son navigateur. L'utilisateur n'est ainsi plus à la merci de la politique de gestion des informations privées des sites cible.

Fondé sur des standards : Shibboleth utilise OpenSAML pour le format des messages et des assertions qui sont basés sur le protocole SAML[4].

[4]Security Assertion Markup Language

Vocabulaire d'attributs standardisés : Shibboleth a définit un ensemble d'attributs standards, comme par exemple la classe d'objet eduPerson largement utilisé dans les milieux universitaires.

Shibboleth est programmé principalement en Java sous forme de servlets. Il nécessite donc l'intsallation d'un serveur Tomcat au dessus d'un serveur web (Apache pour l'occasion). Le travail qui est à réaliser est donc d'installer toutes les briques logicielles nécessaire au bon fonctionnement de Shibboleth, installer NoCat, et l'adapter son système d'authentification pour y rajouter Shibboleth.

3.2.5 Journalisation

La récupération d'informations est faite à plusieurs niveau de l'infrastructure. Au final, c'est une base de donnée PostgreSQL qui stocke l'information ainsi qu'un fichier à plat pour les informations de suivi de connections. Ce que l'on va stocker dans la base est :

Base de donnée

uid : l'identifiant de l'utilisateur, son "login".

orig : origine de l'utilisateur, son université. Par exemple l'utilisateur s'étant authentifié avec le login `mgeli` et ayant choisi l'université Bordeaux 1 aura un uid égal à `mgeli` et une orig de `u-bordeaux1.fr`.

mac : adresse physique du client.

ip : adresse IP qui a été attribuée au client par DHCP

gwid : adresse IP de la passerelle NoCat.

ap_id : identificateur du point d'accès sur lequel le client est connecté

time_in : heure à laquelle s'est authentifié avec succès l'utilisateur.

time_out : heure à laquelle s'est déconnecté l'utilisateur

kb_in : quantité de données en kilo-octets que l'utilisateur à reçu à l'instant t.

kb_out : quantité de données en kilo-octets que l'utilisateur a envoyé à l'instant t.

Suivi de connection

Le fichier texte à plat contenant les informations de connection contient les informations suivantes :

– interface d'entrée

– adresse mac source
– adresse mac destination
– informations d'entête IP
– informations de l'entête de la couche de transport

3.2.6 Accès 802.11i sans portail

3.2.7 Tunneling Ethernet

Le principe revient à encapsuler un paquet ethernet au dessus d'IP. Dans notre cas, on dispose de deux routeurs connectés au niveau IP. Le schéma suivant montre la topologie standard avec nos deux LAN disjoint, et la manière dont ils sont reliés.

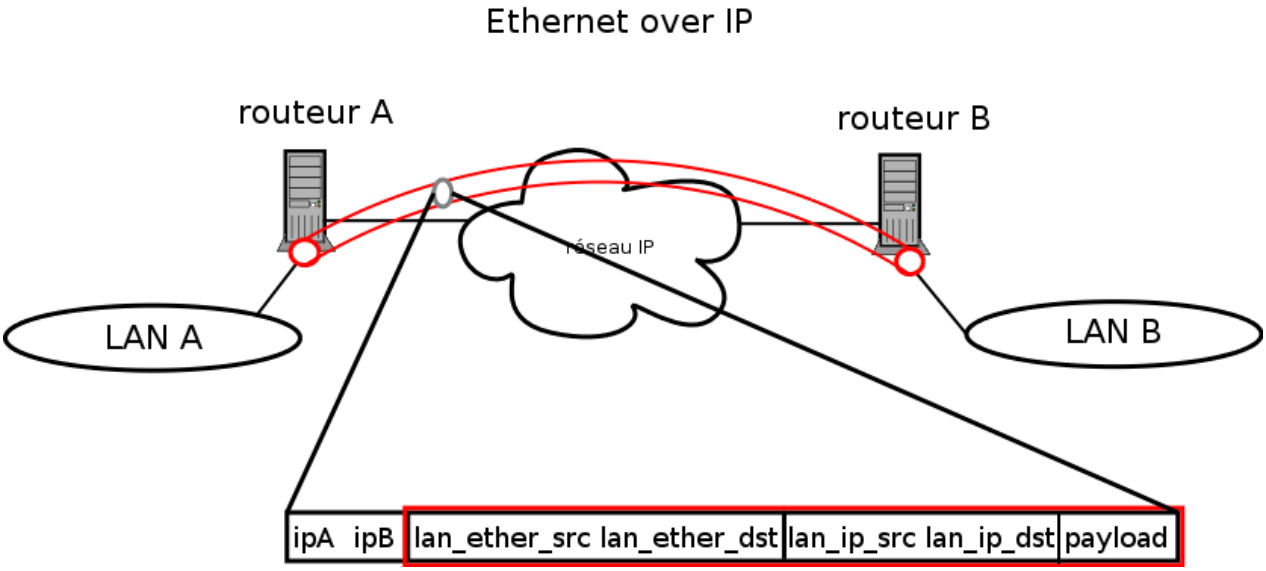

FIG. 3.3 – Tunnel ethernet au dessus d'IP

3.3 Conception détaillée

3.3.1 Shibboleth

Pour pouvoir installer et configurer convenablement shibboleth, il a été nécessaire de bien analyser son fonctionnement. En effet la configuration à

base de nombreux fichiers XML[5] est plutôt fastidieuse. On peut découper Shibboleth en 3 blocs fonctionnels.

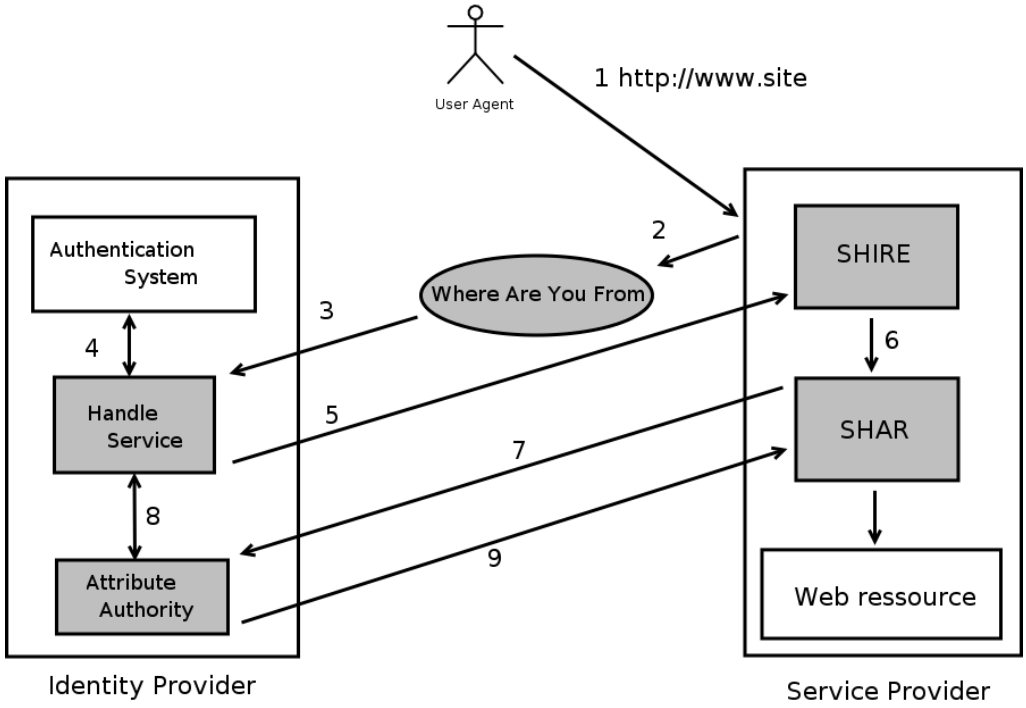

FIG. 3.4 – Flux d'information dans Shibboleth

Service provider : Le fournisseur de service est le serveur que l'on contacte pour accéder à une ressource protégée. Celui ci va alors demander à ce que l'utilisateur s'authentifie par une ressource externe soit en redirigant l'utilisateur en direction du fournisseur d'identité, soit en délégant la tâche de redirection vers le bon fournisseur d'identité au "Where are you from". Ceci étant validé il doit vérifier que l'utilisateur a les bon attributs pour accéder à cette ressource en recontactant le fournisseur d'identité, mais cette fois ci c'est le sous module d'autorité d'attribution qui va être contacté. Ce service est décomposé en deux parties
- le Shar : un serveur indépendant
- le Shire : un module apache, aussi appelé mod_shib.

Where are you from : Cette partie gère la redirection des utilisateurs vers leur université d'origine en leur proposant une liste de toutes les universités valides. Il s'agit d'un servlet Java à part entière. L'utilisateur, ou

[5] eXtented Markup Language

plus précisément son navigateur est redirigé par une redirection HTTP (302).

Identity provider : Fournit 3 sous-modules principaux. Le premier centralise le sous-système d'authentification qui peut exploiter différentes sources d'authentification telles que CAS, LDAP etc...

Le second "Handle Service" est en charge des jetons d'authentification, il les génere après une authentification réussie ou informe si le jeton n'est plus/pas valide. Et finalement le dernier module

3.3.2 Intégration

L'interaction avec un programme externe qui voudrais utiliser Shibboleth se fait au niveau du Shire. Celui-çi fournit aux scripts CGI, par le biais des variables d'environnements, ses propres variables spécifiques. Par exemple, lorsque une personne s'est authentifiée, on peut configurer shibboleth pour placer dans la variable `$REMOTE_USER` l'identificateur de l'utilisateur, que l'on pourra ensuite exploiter à partir de notre CGI afin de vérifier que cet utilisateur est bien prévu pour accéder à ce type de contenu, journaliser l'évènements etc...

Lorsqu'on dispose d'un serveur web diffusant du contenu, et que l'on veut contrôler l'accès à ce contenu en utilisant shibboleth, il suffit de mettre en place un fournisseur de service, et d'indiquer au serveur web (apache) quel répertoire est à "sécuriser". Ensuite on configure notre Shire pour soit renvoyer l'utilisateur sur un aiguilleur (Where are you from) ou directement sur un fournisseur d'identité. partenaires.

Plus précisément dans */etc/apache/httpd.conf* :

```
<Location /secured_dir>
AuthType shibboleth
ShibRequireSession On
require valid-user
</Location>
```

Un script CGI Perl simple a été fait pour vérifier que les attributs sont correctement transmis entre le fournisseur d'identité et le fournisseur de service. Ce script est à placer dans le répertoire sécurisé. On lui fait afficher toutes les variables d'environnement disponible :

```
#! /usr/bin/perl
print "Content-type: text/html\n\n";
foreach $e (sort keys %ENV)
{
```

```
    print "$e -> $ENV{$e}<BR>\n";
  }
```

On voit alors apparaître si tout se passe bien de nouvelles variables associées à leurs valeurs. Le nom des variables spécifique Shibboleth sont préfixées par `HTTP_SHIB` telle que `HTTP_SHIB_ORIGIN_SITE` qui décrit le nom du fournisseur d'identité.

3.3.3 NoCat

Le fonctionnement interne de NoCat n'a quasiment pas été modifié. Le travail a plus porté sur l'adaptation et ajouts de modules externes tels que :

authentification : contournement de la procédure de login dans authserv/cgi-bin/login pour rediriger l'utilisateur sur une page sécurisée shibboleth afin de déclencher les mécanismes d'authentification appropriés. S'il y a succès, on récupere les informations utilisateur en vue de la journalisation.

traduction : françisation des pages par liens symboliques dans authserv/htdocs/

export d'informations vers base : encodage et envoi de la requete HTTP pour les logs. Création du module perl gw/lib/NoCat/Postgre.pm, et modification de Firewall.pm

suivi de connection : adaptation du script d'activation iptables : gw/bin/access.gw en rajoutant plus d'informations pour les logs de connections (ulogd) modification également de Firewall.pm à cet effet.

script serveur d'envoi vers la base : réception de la requête HTTP de log. Script CGI crée : /var/www/nocat/rcv_pg_log.pl

tableau de bord : Mis en forme des logs à disposition sur une page web : /var/www/nocat/pg/dump_pg.pl, dump_sum_pg.pl, stat_nocat.pl. Elle est protégée par mot de passe (/etc/apache/nocat.db). Dans le futur "shibbolethiser" cet accès.

écran de bienvenue : Vérification si premier login de la journee avec affichage d'un splash : /var/www/nocat/splash.pl (ajout de conf dans gw/nocat.conf)

3.3.4 Journalisation

La journalisation des informations se découpe en 3 modules. En effet il faut pouvoir gérer plusieurs évènements de nature différents.

Assocation : Cela se produit lorsque la liaison entre un client et un point d'accès a été établie. C'est un équivalent à une connection physique sur le port d'un switch. L'association est libre dans le cas du portail captif. Lors de cette association la borne émet un message vers un serveur syslog en donnant les informations suivantes : son ip (potentiellement résolue en nom), l'heure de l'évènement, le type de message (association, désassociation, autre), l'interface sur laquelle elle s'est effectuée, l'adresse MAC du client. Un exemple est :

```
Jul  8 18:49:22 soucoupew1.niveau2.u-bordeaux.fr 273: *Mar 16
04:00:03.354: %DOT11-6-ASSOC: Interface Dot11Radio0, Station
      %0011.5015.0b53 Associated KEY_MGMT[NONE]
```

Authentification : Lorsqu'un client est connecté au portail captif après que son couple identifiant, mot de passe ait été accepté, on a l'évènement authentification qui déclenche un log. Le code qui fait ça a été rajouté dans la gateway, ainsi qu'un script CGI sur le serveur web qui fait l'intermédiaire entre nocat et la base postgresql. La communication entre le bout de code côté nocat et le CGI se fait par une requête HTTP. Afin d'assurer la validité des informations et leur origine on rajoute aux informations passées un hash md5 contenant un mot de passe partagé entre les deux éléments concaténé aux informations précédentes :

```
secret = m0td3p4ss3
string = action + time + ip + mac + uid + orig + gw
hash = md5(string . secret);
envoyer(string + hash)
```

Le script CGI qui reçoit ces informations n'a plus qu'a calculer le md5 des arguments en clair envoyés concaténés avec le secret partagé et comparer avec le hash livré en fin de chaîne.

3.3.5 802.11i

L'évolution que l'on peux imaginer dans un futur proche est que tout les utilisateurs seront équipé de matériel compatible 802.11i et qu'il existe sur tout les système une gestion décente de cette technologie. Dés lors on pourra proposer une alternative plus sécurisée au portail captif pour ce qui est de l'accès web. La partie contrôle d'accès réalisée par shibboleth n'étant pas remise en cause. 802.11i est une technologie qui permet de sécuriser 802.1x en rajoutant un mécanisme de rotation de clef et dans le futur utilisera exclusivement AES pour chiffrer le trafic sans variation de clef. Ne disposant pas du matériel compatible 802.11i, les tests effectués ont utilisé la technologie WPA, un sous-ensemble de 802.11i.

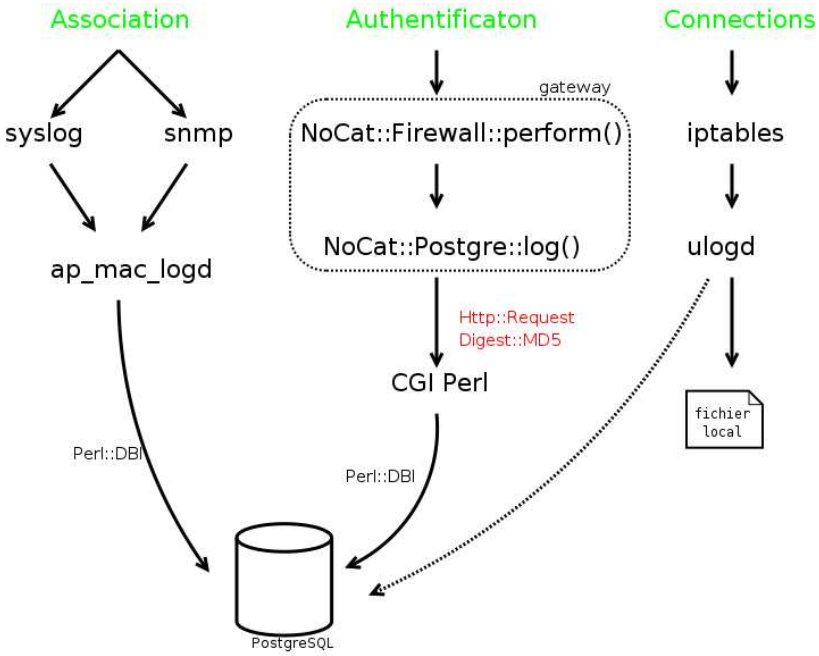

FIG. 3.5 – Système de journalisation

3.3.6 Accès au réseau par authentification

Authentification à partir d'un radius local

Dans un premier temps il a été mis en place un système simple, sans prendre en compte la multiplicité d'origines des étudiants. Les universités stockent les informations sur les étudiants, dont le login et mot de passe dans un serveur LDAP[6]. En premier lieu, la maquette a donc été composé des éléments suivants :

serveur radius : Freeradius

serveur ldap : OpenLDAP

supplicant client : xupplicant, 802.11i côté client Linux, et SecureW2 un client gratuit pour la plateforme Windows.

Voila un schéma explicatif des différents échanges fait lors de la demande de connection au réseau du client. Le contrôle est fait au niveau physique, il y oui ou non association du client sur la borne. Le borne a été configurée pour relayer les informations au serveur radius. Un tunnel est alors crée entre le client et le serveur radius, à l'intérieur duquel se fait l'authentification. A

[6]Lightweight Directory Access Protocol

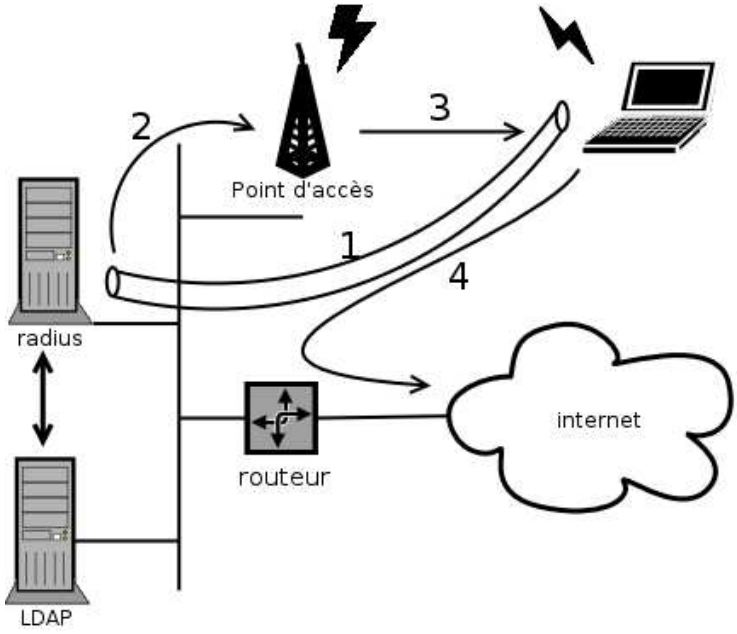

Fig. 3.6 – Authentification avec un radius local

l'issu de cette étape, le serveur radius communique à la borne le verdict, ce qui se solde en une association (ou pas) du client. Celui ci peut alors demander à récupérer une adresse IP par DHCP, et à partir de là commencer à surfer. Pour les tests, le LDAP et le radius étaient en fait sur la même machine.

Un point qui a été volontairement omis est la manière dont le client s'est authentifié. En effet, il y a plusieurs choix possible d'authentification suivant le niveau de sécurité dont on dispose ainsi que les ressources. Ce qui a été testé est :

EAP/MD5 : authentification utilisant CHAP, utilisé à des fin de tests uniquement du serveur radius. Déconseillé car les hashes sont transmis en clair, et ne disposent pas d'une authentification mutuelle.

EAP/TLS : Authentification mutuelle par des certificats X.509. Il a donc été nécessaire de générer des certificats pour chaque entité avec les outils fournis par OpenSSL. Pour les tests il y avait un nombre limité de certificats à gérer mais à l'échelle universitaire la gestion d'une PKI[7] commence à être lourde à gérer.

EAP/TTLS : Authentification TLS à l'intérieur d'un tunnel TLS établie entre le client et le serveur radius. Dans ce cas la PKI est à gérer que du

[7]Public Key Infrastructure

côté du serveur. Solution envisageable dans un contexte universitaire où l'on mettrais à disposition des étudiants su un intranet par exemple, le certificat serveur.

Toutes ces solutions ont été validées avec succès en environnement de test, la dernière retenant l'attention pour une future utilisation.

Authentification sur un radius distant par proxification

La phase d'authentification avec un proxy radius, dénote la volonté à s'intégrer dans le futur au projet national d'authentification répartie Arredu en cours de tests.

Voici un schéma tiré du site officiel représentant les différents acteurs d'un tel dispositif :

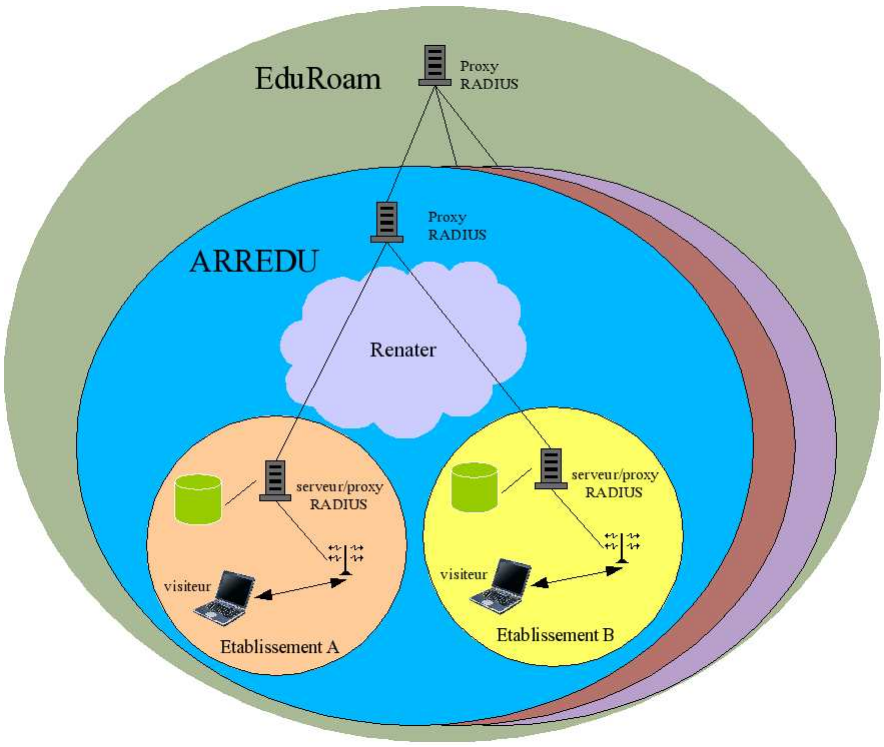

FIG. 3.7 – Infrastructure d'authentification répartie Arredu

Dans ce cadre, les universités sont les cercles de plus petite taille. Le cylindre vert représente une source d'informations, comme un annuaire LDAP par exemple. La politique de Réaumur dans ce cadre n'est pas encore définie,

sachant que le projet en est à ses débuts. Réaumur pourrait proposer à certains établissements n'ayant pas le personnel, ou les moyens pour gérer leur propre serveur radius de le faire à leur place. Cependant cela pose quelques problèmes sachant que même si les communications sont sécurisées entre les divers éléments du dispositif, le radius Réaumur quand il n'agit pas en tant que proxy a à un moment entre ses mains les mots de passe utilisateur en clair.

Les requêtes d'authentifications sont acheminées vers le serveur d'authentification de l'établissement d'appartenance de l'utilisateur par le biais du nom de domaine (realm) associé à son identifiant (de la forme user@etab.fr). La garantie d'un minimum de fonctionnalités et de sécurité aux utilisateurs exige la définition et le respect d'un ensemble de règles par tous les participants. Cette confiance est formalisée par un engagement des participants auprès de Renater qui s'engage à son tour auprès de Terena[8]

sécurisation de la connection établie

Une fois que l'accès est établie entre le client et la borne il faut sécuriser le lien, qui peut être écouté par n'importe quelle personne mal intentionnée. Les choses se passent donc entre un client WPA (supplicant) situé sur l'ordinateur de l'utilisateur, et la borne. Ce qui a été testé est une rotation régulière de clefs WEP de 128bits. Les supplicant WPA testés que ce soit sous Linux ou Windows XP ont bien supporté la rotation de clef gérée par les bornes Cisco. Cette solution basé sur des clef WEP[9] tournante est assez intéressante dans le sens ou tout matériel wifi peut le supporter à condition de disposer du bout de code client (supplicant). Avec une période relativement faible telle que 1 minute, casser une clef WEP 128 bit en moins d'une minute demande une puissance de calcul actuellement non envisageable.

3.3.7 Tunneling Ethernet

Faire un tunnel de niveau 2 entre deux LAN a un interêt lorsque l'on a des partenaires distant pour mutualiser les ressources du portail captif. En effet ces partenaires sont reliés par des liaisons spécialisés au niveau 3 sur les équipements de Réaumur. Du coup, on ne peux pas router les adresses privées des clients wifi. Un tunnel de niveau 2 nous permet de faire passer les requêtes de demande d'adresse IP DHCP. Les outils qui permettent de faire ça principalement sont VTun et openvpn. La simplicité de mise en oeuvre, la légèreté, et sa communauté active font d'openvpn un candidat de choix.

[8]Trans-European Research and Education Networking Association
[9]Wired Equivalent Privacy

Mettre en place un tunnel L2 nécessite de disposer à chaque extrermité un routeur, avec une instance openvpn.

Les figures suivantes montre la topologie du réseau de test. La première configuration est un test sans faire passer de VLAN dans le tunnel. Les deux tests auraient pu être fait avec la même topologie réseau. C'est juste dans un but didactique qu'a été fait le premier, pour se placer dans une situation non triviale où l'on a conjointement deux tunnels de niveau 2. Le premier étant pour raccorder le réseau MDS (résidence étudiante) avec le routeur côté Réaumur. Et le second permet de prolonger le LAN MDS vers une machine de réaumur qui avait pour l'occasion une interface virtuelle dans ce sous-réseaux.

Sans VLAN

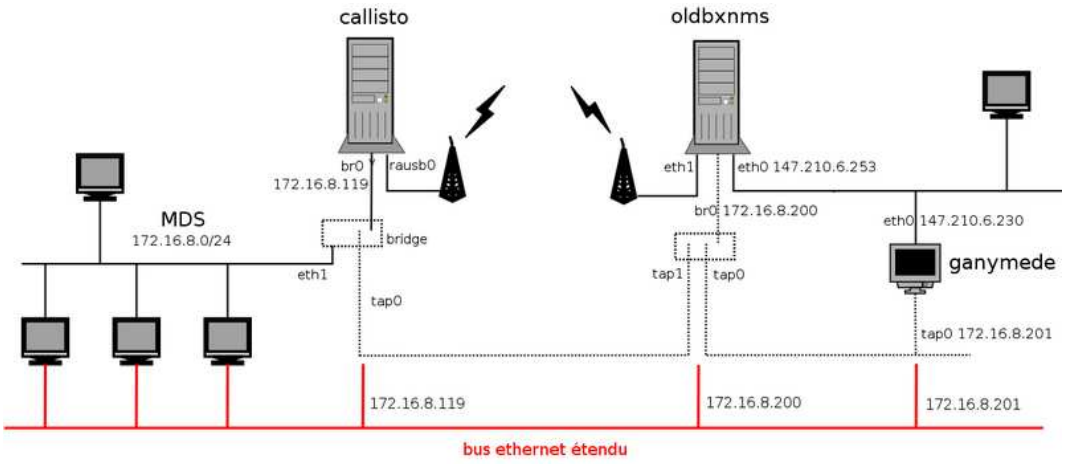

Fig. 3.8 – Raccordement du réseau physique MDS à un réseau virtuel mds côté Réaumur

Cette idée peut être appliquée pour un LAN wifi distant d'un partenaire. Mais il faudrait créer un tunnel pour chaque VLAN véhiculé. Ce qui est réalisable, mais peu pratique.

Avec VLAN

L'étape suivante a donc été la recherche d'un raccordement de plusieurs VLAN dans le même tunnel. Cela peut en fait être réalisé avec la technique de port trunking qui revient à faire passer sur la patte côté wifi du routeur

des paquets tagués avec leur numéro de VLAN. Cette configuration se fait à plusieurs endroits :

- au niveau des bornes afin qu'elles marquent bien les paquets.
- sur tout les commutateurs que l'ont peut trouver entre le routeur et les bornes afin d'assurer la continuité des VLAN.
- configurer le trunk sur le lien du routeur
- notre routeur doit pouvoir comprendre ces paquets un peu spéciaux. Il faut donc activer le support 802.1Q dans le noyau Linux.

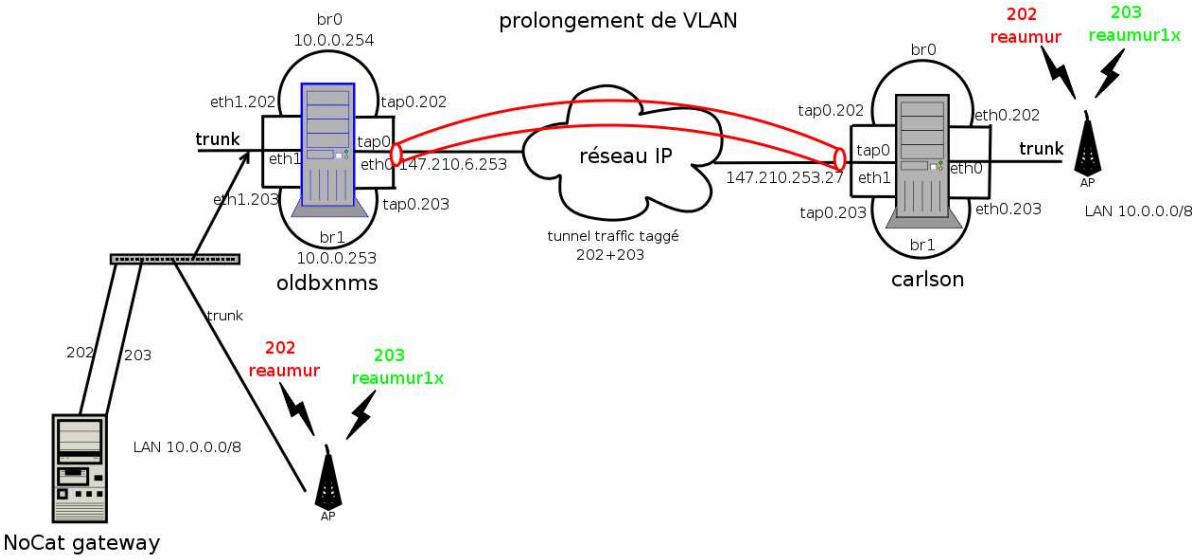

FIG. 3.9 – tunnel ethernet avec VLAN vers un partenaire

3.4 Tableau de bord

Un tableau de bords a été développé dans le but d'offrir à l'administrateur une vision globale du système et de pouvoir détecter un abus, ou une panne. Il y a plusieurs pages :

- informations de connection en temps réel, reprenant les valeurs énnoncées en 3.2.5
- historique des informations précedentes avec choix dans la plage de temps.
- affichage des règles du firewall, pour les tests uniquement.

De plus a été intégré en fin de stage un outil qui permet par un simple bouton sur le tableau de bords de contrôler le débit de chaque utilisateur

(adresse IP). Ceci permet de se prémunir des problèmes de déni de service initiés pouvant être initiés par certains vers ou autre malware. On peux imaginer aussi un comportement hors charte des utilisateurs face au Peer 2 Peer. Effectivement il est désormais facile de faire passer par du traffic web des logiciels du type kazaa, emule...

L'outil utilisé pour développer cette solution est *tc* de la suite logicielle *iproute2*. L'interfaçage web est fait par un script CGI shell. La syntaxe de tc est assez spéciale, et non triviale, mais le support fournis par le canal IRC, et l'opportunité de pouvoir discuter avec son développeur principal on été un plus. L'explication du fonctionnement de tc et de ses concepts sous-jacents sortirais du cadre de ce rapport.

Chapitre 4

Conclusion

L'objectif des différentes parties du stage a été atteint. Il reste maintenant à maintenir le système en l'état dans un environnement plus hostile avec la rentrée étudiante sous peu. Le suivi soutenu durant ces 3 mois par mon maître de stage ne peut que rendre la transition plus douce et sans surprise. Le caractère fortement orienté administration réseau et sécurité de ce stage m'a permis de toucher à un maximum de technique dans des domaines qui devraient m'être familier dans un futur proche. En effet que ce soit pour ma fin de scolarité à l'ENSEIRB en filière RSR, ou en stage au CEA je devrais être amené à manipuler ces concepts.

Le matériel évoluant on peux imaginer que dans un futur proche tout les équipements soient compatibles avec la norme 802.11i. Dans ce cas, le portail captif n'aurait plus vraiment lieu d'être et l'évolution logique serait de se recentrer sur une solution à base de radius, et de s'intégrer à une fédération d'identité inter-universitaires nationale, voire internationale.

Chapitre 5

Annexes

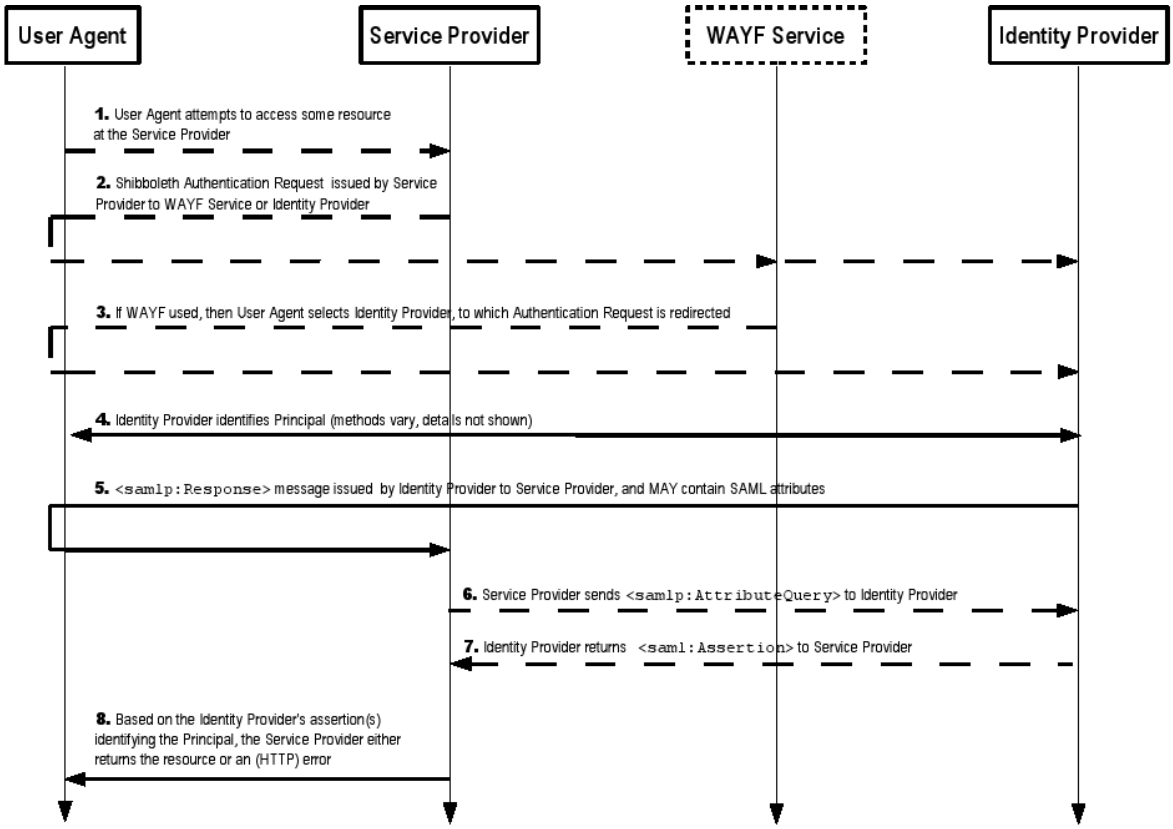

FIG. 5.1 – Diagramme de séquence de Shibboleth

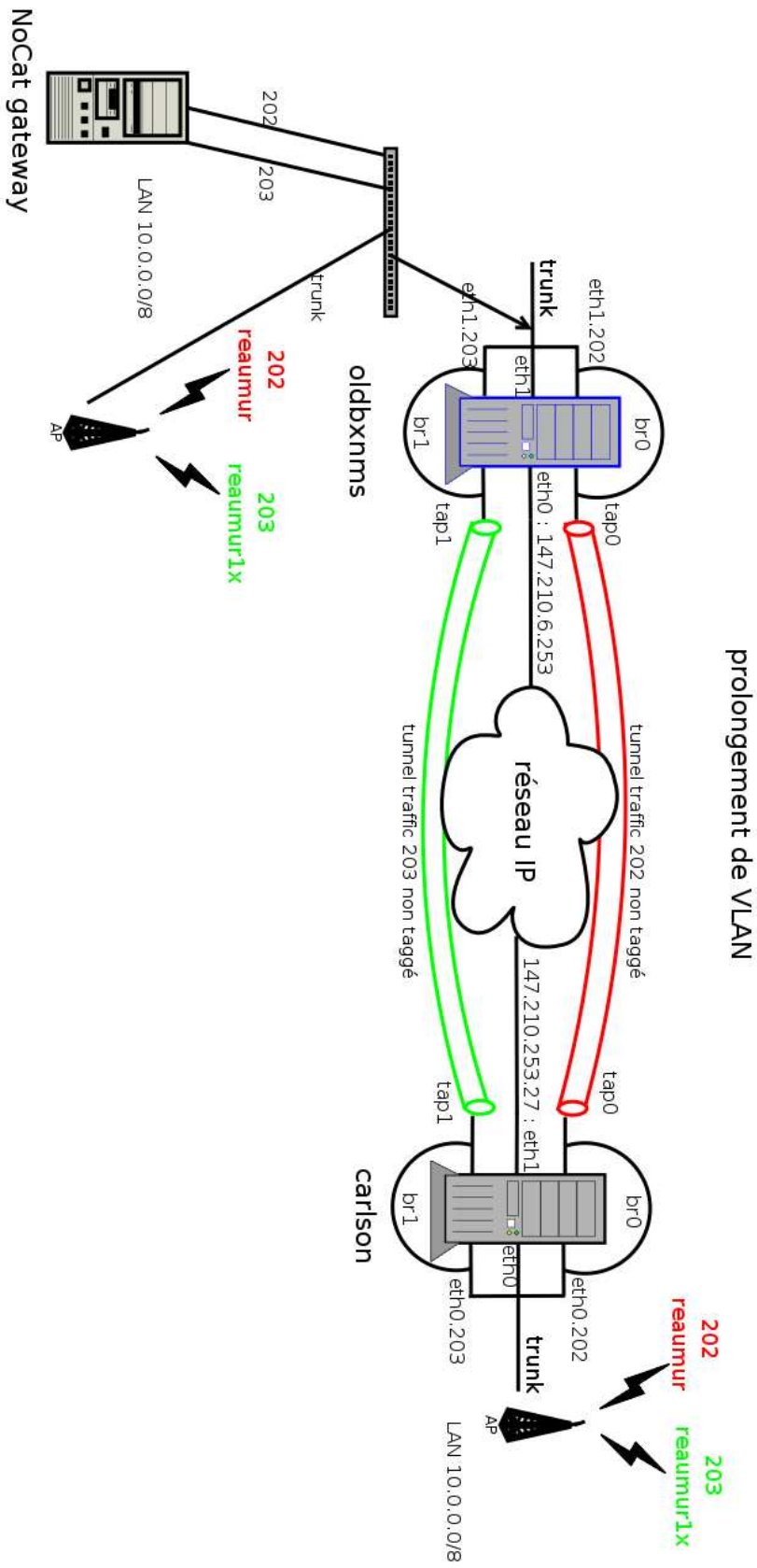

FIG. 5.2 – Tunnels OpenVPN par VLAN

5.0.1 Configurations IOS

5.0.2 Côté switch

Le switch a un trunk sur le port FastEthernet0/1 la borne sera connectée sur le port FastEthernet0/5 La continuité des VLANS est réalisée par le switch en amont.

Définition des VLANs de la borne :

```
!
vlan 201
 name wifi stage
!
vlan 202
 name vlan client borne wifi
```

Le port de trunk a un vlan natif de 29 (reaustage). Pas de configuration spécifique sur ce port.

```
!
interface FastEthernet0/1
 switchport trunk native vlan 29
 switchport mode trunk
 spanning-tree portfast
!
```

Le port de la borne laisse passer les vlans 201 à 203, et celui d'admin (201) est le natif.

```
! interface FastEthernet0/5

description borne wifi stage
switchport trunk native vlan 201
switchport trunk allowed vlan 201-203
switchport mode trunk
switchport nonegotiate
spanning-tree portfast

!
```

Si l'ont veut connecter des clients wifi en filaire sur le switch, il suffit de les placer dans le VLAN 202 :

```
!
interface FastEthernet0/6
 switchport access vlan 202
!
```

5.0.3 Côté borne

La borne 1130 a plusieurs interfaces physiques à configurer :
- inteface wifi B/G : **Dot11Radio0**
- interface wifi A : **Dot11Radio1**
- interface ethernet : **FastEthernet0**

Ensuite on peux configurer des sous-interfaces correspondant aux VLANs que l'on a définis : ex : Dot11Radio0.201 va désigner le VLAN 201 de l'interface wifi B/G.

Création des différents VLANs

dot11 vlan-name ADMIN-WIFI vlan 201 dot11 vlan-name PORTAIL-CAPTIF vlan 202 dot11 vlan-name WIFI-802.1X vlan 203

Association d'un SSID à un VLAN

```
!
dot11 ssid REAUMUR
   vlan 202
   authentication open
!
```

Association de plusieurs SSID à une interface

```
interface Dot11Radio0
 description Wireless BG
 no ip address
 no ip route-cache
 !
 ssid REAUMUR
 !
 ssid REAUMUR1X
 !
```

Configuration des sous-interaces (classées par VLAN)

(exemple du VLAN 202 de l'interface radio B/G)

```
!
interface Dot11Radio0.202
 encapsulation dot1Q 202
 no ip route-cache
 bridge-group 202
 bridge-group 202 subscriber-loop-control
 bridge-group 202 block-unknown-source
 no bridge-group 202 source-learning
 no bridge-group 202 unicast-flooding
 bridge-group 202 spanning-disabled
!
```

(puis pour l'interface ethernet)

```
!
interface FastEthernet0.202
 encapsulation dot1Q 202
 no ip route-cache
 bridge-group 202
 no bridge-group 202 source-learning
 bridge-group 202 spanning-disabled
!
```

Bridging entre les interfaces Wifi et Ethernet

exemple du pont entre le wifi B/G, VLAN 202 (utilisateurs wifi non 802.1x) et l'interface Ethernet0 : L'exmple ci-dessus nous montre comment bridger les interfaces Dot11Radio0.202 et FastEthernet0.202 en utilisant le bridge-group commun 202.

Pour les même interfaces, VLAN 201, on remarquera que c'est le bridge-group 1 d'utilisé et que dans la conf globale est rajouté la directive :

```
bridge 1 route ip
```

Ce qui nous permet de nous connecter en IP sur la borne dans le VLAN d'administration.

Désactivation du bridging fait entre Dot11Radio0.201 et Ethernet0.201

```
geliw1#config t
geliw1(config)#no interface dot11Radio 0.201
```

Ca a l'air de bien marcher sans, et il n'y a pas de raisons que ce bridge soit fait.

5.0.4 Comportement hasardeux IOS

Apparement si l'on crée un vlan en global et qu'on le recrée dans une interface, le global ne peux pas être effacé. Il a fallu sauvegarder la config actuelle, et faire un reset de la borne. Pour tout recharger à la main à partir du backup.

Faire des modifications à partir de la CLI *et* de l'interface web peux provoquer des comportements aléatoires.

5.1 Configuration du ldap

Cette doc traite de slapd, serveur du projet libre OpenLDAP d'implémentation ldap sous GNU/Linux. Lors de la config debian de ldap on a spécifie le nom de domaine suivant : ganymede.u-bordeaux.fr Ca correspondra en terme LDAP à dc=ganymede,dc=u-bordeaux.fr,dc=fr
On peut voire la ',' comme le séparateur de répertoire UNIX '/' sauf qu'en LDAP pour lire de la racine à la feuille il faut lire de droite à gauche.

5.1.1 Création d'un utilisateur

On veux se créer juste en dessous de ganymede l'utilisateur 'Mathieu GELI' :

```
dn: cn=Mathieu GELI,dc=ganymede,dc=u-bordeaux,dc=fr
objectClass: inetOrgPerson
cn: Mathieu GELI
givenName: Mathieu
sn: GELI
uid: mgeli

$ ldapadd -W -v -x -h 147.210.6.230 \
  -D "cn=admin,dc=ganymede,dc=u-bordeaux,dc=fr" -w admin -W -f add\_user.ldif
```

5.1.2 Création d'un sous-arbre

ou : organizationUnit On veut se créer une classe de personnes juste en dessous de la racine ganymede.u-bordeaux.fr.

On crée le fichier suivant :

```
dn: ou=people,dc=ganymede,dc=u-bordeaux,dc=fr
objectclass: organizationalUnit
ou: people
```

```
$> ldapadd -W -v -x -h 147.210.6.230 \
   -D "cn=admin,dc=ganymede,dc=u-bordeaux,dc=fr" -w pass -f subtree.ldif
```

5.1.3 Modification : ajout

Ajout d'un champs dans un enregistrement existant : On crée le fichier mod_user.ldif suivant :

```
dn: cn=Mathieu GELI,dc=ganymede,dc=u-bordeaux,dc=fr
changetype: modify
add: uid
uid: mgeli
```

```
$> ldapadd -W -v -x -h 147.210.6.230 \
  -D "cn=admin,dc=ganymede,dc=u-bordeaux,dc=fr" -w pass -f mod_user.ldif
ldap\_initialize( ldap://147.210.6.230 )
Enter LDAP Password:
add uid:
      mgeli
modifying entry "cn=Mathieu GELI,dc=ganymede,dc=u-bordeaux,dc=fr"
modify complete
```

5.1.4 Modification : retrait

On crée un fichier contenant :

```
dn: cn=Mathieu GELI,dc=ganymede,dc=u-bordeaux,dc=fr
changetype: delete
```

```
$> ldapadd -W -v -x -h 147.210.6.230 \
   -D "cn=admin,dc=ganymede,dc=u-bordeaux,dc=fr" -w pass -f del\_user.ldif
```

5.1.5 Récupération d'attributs

Un client qui a un acces au LDAP peux emettre des requêtes pour récupérer des attributs suivant des critères. On peux même se servir d'opérations booléennes.

Tout d'abord on veux lister en temps qu'utilisateur anonymous (non authentifié) les attributs de 'Mathieu GELI' :

anonyme

```
$> ldapsearch -x -LLL -h 147.210.6.230 -b \
   'dc=ganymede,dc=u-bordeaux,dc=fr' 'uid=mgeli'

dn: cn=Mathieu GELI,dc=ganymede,dc=u-bordeaux,dc=fr
objectClass: inetOrgPerson
cn: Mathieu GELI
givenName: Mathieu
sn: GELI
uid: mgeli
```

authentifiée

On voudrais s'authentifier pour pouvoir récupérer notre mot de passe. On va donc se binder sur l'utilisateur Mathieu GELI (les ACL permettent aux users authentifiés d'accéder à leur champs userPassword)

```
$> ldapsearch -wpass -x -LLL -h 147.210.6.230          \
   -D 'cn=Mathieu GELI,dc=ganymede,dc=u-bordeaux,dc=fr' \
   -b 'dc=ganymede,dc=u-bordeaux,dc=fr' 'uid=mgeli' userPassword

dn: cn=Mathieu GELI,dc=ganymede,dc=u-bordeaux,dc=fr
userPassword:: dGVzdA==

(c'est le hash de 'test')
```

booléenne

```
$> ldapsearch -x -h localhost -b "dc=ceenet,dc=ceu,dc=hu" \
   "(&(objectclass=person)(ou=accounting))"
```

La syntaxe est relativement directe, on commence par noter l'opérateur, puis on accole entre parenthèse ces arguments. Il s'agit de notation préfixe (ou polonaise inversée) (à adapter pour ganymede)

5.1.6 Création d'attributs

On crée un fichier **wifiperson.schema** dans le répertoire */etc/ldap/schema/* :

```
attributetype ( 0.9.2342.19200300.100.1.100 NAME 'wifiAccess'

DESC 'say if a user is allowed to connect'

        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} )
```

cette syntaxe définit un nouvel attribut avec ces différentes caractéristiques, le nommage 0.9.2342.19200300.100.1.100 a été pris un peu au hasard, de manière à ce que ce ne soit pas occupé par les attributs définis dans le répertoire en question. Il faudrait voire à quoi cela correspond au niveau de la MIB. Le champ SYNTAX définit le type, ici une chaîne de caractère de 256 caractères maximum. Toujours pareil, c'est par un nommage à la SNMP qu'on décrit le type.

Pour prendre en compte ce fichier lors du démarrage du démon slapd il faut rajouter dans */etc/ldap/slapd.conf* :

```
# Schema and objectClass definitions
include         /etc/ldap/schema/core.schema
include         /etc/ldap/schema/cosine.schema
include         /etc/ldap/schema/nis.schema
include         /etc/ldap/schema/inetorgperson.schema
*** include   /etc/ldap/schema/wifiperson.schema</div> ***
```

Il est possible de faire de l'objet et d'avoir des relations d'héritage entre attributs. Dans ce cas, l'ordre dans lequel ils apparaîssent dans le fichier de configuration importe.

on relance notre démon :

```
# /etc/init.d/slapd restart
```

et l'on vérifie que tout marche bien, en rajoutant ce nouvel attribut à 'Mathieu GELI' par exemple :

```
dn: cn=Mathieu GELI,dc=ganymede,dc=u-bordeaux,dc=fr
changetype: modify
add: wifiAccess
wifiAccess: ok
```

```
$ ldapadd -W -v -x -h 147.210.6.230              \
  -D "cn=admin,dc=ganymede,dc=u-bordeaux,dc=fr" \
  -w admin -W -f subtree.ldif

ldap\_modify: Object class violation (65)
    additional info: attribute 'wifiAccess' not allowed
```

Et là, on, se trouve face à cette erreur. Je n'ai pas investigué bien longtemps, mais il semble que la définition de notre attribut n'es pas correcte d'un point de vue hiérarchique. Il faudrait probablement, du fait que notre utilisateur soit un object **inetOrgPerson**, faire dériver l'attribut wifiAccess de cet objet. Neammoins le fait de mettre dans slapd.conf :

```
schemacheck     off
```

permet de résoudre ce problème de violation. A mettre au clair!!

5.1.7 Références

- Browser Java (http ://www-unix.mcs.anl.gov/ gawor/ldap/index.html)
- Un tuto assez général (http ://www.ceenet.org/workshops/lectures2001/Peter%20Gietz/tut (en)
- tuto français simple, moins général (http ://www.coagul.org/article.php3 ?id_article=172)
- Cours sur LDAP (http ://www.commentcamarche.net/ldap/ldapintro.php3)
- Définitions de ses propres atttributs

5.2 Mise en place du radius+supplicant

5.2.1 Brêve intro

http ://www.freeradius.org/
Implémentation de serveur Radius libre.

Le principe étant de permettre à un client d'accéder à des ressources réseau au travers d'un NAS (Network Access Server). Celui ci relâchant les ressources en fonction de la décision du serveur Radius.

Le protocole utilisé pour cette mise en place étant un des plus sécurisé actuellement est EAP-TLS.

5.2.2 EAP/MD5

Il n'y a rien de crypté dans ce protocole ppp-like. Pour tester notre radius on se mets en authentification par comparaison de hashs md5. Le serveur a le

mot de passe client stocké quelque part et compare le md5 que le supplicant lui donne avec le sien.

Tout d'abord on installe freerdius :

```
# apt-get install freeradius
```

Puis dans le fichier users on configure les login/pass des utilisateurs et le type d'authentification (facultatif)

```
matth Auth-Type := EAP , User-Password == "testpassword"
```

La borne wifi doit être configurée pour passer par le Radius. Sur la Aironet 1100, on se mets en open authentication et on selectionne with EAP

5.2.3 Côté client

Xsupplicant : http ://www.open1x.org
Implémentation libre de 802.1X côté client. Prise en charge des différentes méthodes EAP d'authentification.

```
\# apt-get install xsupplicant
```

la conf se fait assez simplement pour une authentification par EAP-MD5 (danger sur du sans-fil!!)

```
network_list = tsunami
default_netname = tsunami
allow_interfaces = wlan0
deny_interfaces = dummy0, eth0, sit0

tsunami
{
   allow_types = eap_md5
     eap-md5 {
       username = <BEGIN_UNAME>testuser<END_UNAME>
       password = <BEGIN_PASS>testpassword<END_PASS>
     }
}
```

5.2.4 wpa supplicant

http ://hostap.epitest.fi/wpa_supplicant/ Un autre supplicant WPA sous Linux, non testé.

5.2.5 EAP/TTLS

Le principe d'EAP-TLS est que le client et le serveur s'authentifient mutuellement par le moyen de certificats client ET serveur. A partir de là, chacun est assuré de l'identité de l'autre. L'usuel couple login/password est donc avantageusement remplacé ici par certif client/certif serveur/certif racine (pour vérifier que le serveur est bien signé par une autorité de certification trustée). Avantageusement, parceque ce moyen d'authentification est un des plus robuste actuellement vu qu'il repose sur la crypto forte (asymétrique), par contre ça necessite un tout petit plus de configuration côté client que d'habitude...

Pour TTLS il s'agit juste d'un certif niveau serveur. L'identité du client n'est donc pas vérifiée. Dans certain cas, cette méthode est satisfaisante, quand on a tout un campus à gérer, ou un acces publique.

Bon, pour commencer je suis obligé de compiler à partir des sources freeradius car il n'y a pas la librairie dynamique rlm_eap_tls.so dans la version debian (1.0.4-2). Problèmes de politique.

La version debian d'openssl est la 0.9.7e-3.

compilation

```
./configure --prefix=/usr --sysconfdir=/etc
./make
./checkinstall make install (création d'un paquet debian + installation)
```

Configuration

radiusd.conf

```
modules {
      ...
      pap {
            encryption_scheme = clear
      }
      ...

}

authorize {
 ...
 eap
 ...
```

```
}
authenticate {

       Auth-Type PAP {
              pap
       }
       ...
       eap
}
```

eap.conf

```
tls {
  private_key_password = plop
  private_key_file = /etc/freeradius/certs/cert-srv.pem
  certificate_file = /etc/freeradius/certs/cert-srv.pem
  CA_file = /etc/freeradius/certs/root.pem
  dh_file = /etc/freeradius/certs/dh
  random_file = /etc/freeradius/certs/random
  fragment_size = 1024
  include_length = yes
}
```

Oui, effectivement il faut configurer TLS bien que l'on fasse du TTLS, mais le client ne pourra pas faire de TLS bien entendu (il n'as pas de certif client, donc n'ira pas bien loin dans la transaction). Le certificat serveur dont il est question ici est censé avoir été généré par des commandes openssl standard donc j'en fait une petite synthèse plus loin.

```
ttls {
  default_eap_type = md5
  copy_request_to_tunnel = no
  use_tunneled_reply = yes
}
```

users

Dans ce fichier on définit localement les utilisateurs pouvant être authentifiés, les méthodes d'authentification ainsi que s'il est nécessaire, leur mot de passe associés. Nous faisons un test avec mot de passe en clair dans le fichier users (Local) :

On commente donc toutes les lignes de users, et on rajoute en fin de fichier ceci :

```
mgeli Password =="test"
```

Remarque : ce nom d'utilisateur (mgeli) est matché avec celui de la balise pap dans le supplicant client car on a mis `copy_request_to_tunnel = no` dans eap.conf, sinon il aurait été matché avec le champ `identity` du supplicant.

Freeradius va alors ballayer les différents choix d'authentification qu'il a, et ne trouvera (on a commenté tout les autres) que le Auth-Type := LOCAL il n'est donc pas nécessaire de le renseigner.

Côté client (supplicant)

Xsupplicant / Linux

xsupplicant.conf

Dans le bon SSID on mets :

```
identity = <BEGIN_ID>mgeli@bordeaux1<END_ID>
allow_types = eap_ttls
eap-ttls {
   root_cert      = /etc/xsupplicant/tls/ca_cert.pem
   phase2_type = pap
   pap {
     username = <BEGIN_UNAME>mgeli<END_UNAME>
     password = <BEGIN_PASS>test<END_PASS>
   }
}
```

Voilà, ici la directive rootcert est un chemin vers le certificat de l'autorité racine, ceci permet au client de vérifier que le serveur auquel il se connecte (qui lui envoie son certificat durant l'échange TLS) est bien un certificat signé par cette même autorité. Ceci afin d'éviter les attaques MITM (Man in the middle). Il faut donc transférer par un canal sûr (SSH, HTTPS) ce ca_cert.pem du serveur vers le client.

On monte l'interface, lance xsupplicant, et fait une requête DHCP (ordre important).

```
\# ifconfig wlan0 up
\# xsupplicant -i wlan0 -c /etc/xsupplicant/xsupplicant.conf -D wext
\# dhclient wlan0

\# ping 10.0.0.254
64 bytes from 10.0.0.254: icmp_seq=1 ttl=64 time=1.26 ms
```

SecureW2 / Windows XP

Client gratuit, et mieux intégré à Windows que Aegis (payant !). Suivre les indications du lien suivant, en particulier pour l'import du certificat racine dans le bon magazin.

http ://cri.univ-mlv.fr/wifi/html/conf-802.1x-windows-XP.html#config-secureW2

test du serveur

On lance celui-ci en mode debug :

```
 radiusd -X -A
 [...]
 Module: Loaded eap
  eap: default_eap_type = "md5"
  eap: timer_expire = 60
  eap: ignore_unknown_eap_types = no
  eap: cisco_accounting_username_bug = no
 rlm_eap: Loaded and initialized type md5
 rlm_eap: Loaded and initialized type leap
  gtc: challenge = "Password: "
  gtc: auth_type = "PAP"
rlm_eap: Loaded and initialized type gtc
tls: rsa_key_exchange = no
tls: dh_key_exchange = yes
tls: rsa_key_length = 512
tls: dh_key_length = 512
tls: verify_depth = 0
tls: CA_path = "(null)"
tls: pem_file_type = yes
tls: private_key_file = "/etc/freeradius/certs/server_key.pem"
tls: certificate_file = "/etc/freeradius/certs/server_cert.pem"
tls: CA_file = "/etc/freeradius/certs/demoCA/cacert.pem"
tls: private_key_password = "test"
tls: dh_file = "/etc/freeradius/certs/dh"
tls: random_file = "/etc/freeradius/certs/random"
tls: fragment_size = 1024
tls: include_length = yes
tls: check_crl = no
tls: check_cert_cn = "%{User-Name}"
rlm_eap: Loaded and initialized type tls
```

```
ttls: default_eap_type = "md5"
ttls: copy_request_to_tunnel = yes
ttls: use_tunneled_reply = yes
rlm_eap: Loaded and initialized type ttls
peap: default_eap_type = "mschapv2"
peap: copy_request_to_tunnel = no
peap: use_tunneled_reply = no
peap: proxy_tunneled_request_as_eap = yes
rlm_eap: Loaded and initialized type peap
mschapv2: with_ntdomain_hack = no
rlm_eap: Loaded and initialized type mschapv2
Module: Instantiated eap (eap)
Module: Loaded preprocess
preprocess: huntgroups = "/etc/freeradius/huntgroups"
preprocess: hints = "/etc/freeradius/hints"
preprocess: with_ascend_hack = no
preprocess: ascend_channels_per_line = 23
preprocess: with_ntdomain_hack = no
preprocess: with_specialix_jetstream_hack = no
preprocess: with_cisco_vsa_hack = no
Module: Instantiated preprocess (preprocess)
Module: Loaded detail
detail: detailfile = "/var/log/freeradius/radacct/%{Client-IP-Address}/auth-deta
detail: detailperm = 384
detail: dirperm = 493
detail: locking = no
Module: Instantiated detail (auth_log)
Module: Loaded CHAP
Module: Instantiated chap (chap)
Module: Loaded MS-CHAP
mschap: use_mppe = yes
mschap: require_encryption = no
mschap: require_strong = no
mschap: with_ntdomain_hack = no
mschap: passwd = "(null)"
mschap: authtype = "MS-CHAP"
mschap: ntlm_auth = "(null)"
Module: Instantiated mschap (mschap)
Module: Loaded realm
realm: format = "suffix"
realm: delimiter = "@"
```

```
realm: ignore_default = no
realm: ignore_null = no
Module: Instantiated realm (suffix)
Module: Loaded files
files: usersfile = "/etc/freeradius/users"
files: acctusersfile = "/etc/freeradius/acct_users"
files: preproxy_usersfile = "/etc/freeradius/preproxy_users"
files: compat = "no"
Module: Instantiated files (files)
Module: Loaded Acct-Unique-Session-Id
acct_unique: key = "User-Name, Acct-Session-Id, NAS-IP-Address, Client-IP-Addres
Module: Instantiated acct_unique (acct_unique)
detail: detailfile = "/var/log/freeradius/radacct/%{Client-IP-Address}/detail-%Y
detail: detailperm = 384
detail: dirperm = 493
detail: locking = no
Module: Instantiated detail (detail)
Module: Loaded radutmp
radutmp: filename = "/var/log/freeradius/radutmp"
radutmp: username = "%{User-Name}"
radutmp: case_sensitive = yes
radutmp: check_with_nas = yes
radutmp: perm = 384
radutmp: callerid = yes
Module: Instantiated radutmp (radutmp)
Listening on authentication *:1812
Listening on accounting *:1813
Listening on proxy *:1814
Ready to process requests.
```

Paramétrage de la borne

On active pour le VLAN 202 (le 203 ne passe pas pour le moment à travers tout les switchs) :

- le champs `Open Authentication` à `with EAP`
- WEP Encryption : Mandatory
- On spécifie un cycle de clef de 60sec.

Les clefs sont ainsi changées régulièrement par l'AP =¿ gain de sécurité façe aux tentatives de craquages standard. (faiblesse des IV's)

Hack Xsupplicant

```
\# iwconfig 2> /dev/null | grep Encryption
Encryption key:66B6-1016-B31B-62E5-D51C-1C5E-58 [2]   Security mode:open
```

...(60sec plus tard) ...

```
\# iwconfig 2> /dev/null | grep Encryption
Encryption key:6EC1-8098-AEB4-61B8-253B-5BC4-10 [4]   Security mode:open
```

Petit problème rencontré avec Xsupplicant (version CVS) lors du changement de clef on perd la communication. Il faut relancer Xsupplicant pour la retrouver (? ?)

Il serait intéressant de pouvoir vérifier les changements de clef au niveau des logs de la borne. Pour le moment pas trouvé.

SecureW2 (Windows XP)

Suivre les indications du lien suivant, en particulier pour l'import du certificat racine dans le bon magazin. Client gratuit, et mieux intégré à Windows que Aegis (payant !).

http ://cri.univ-mlv.fr/wifi/html/conf-802.1x-windows-XP.html#config-secureW2

5.2.6 Proxy radius

Monter un proxy radius n'est pas trop compliqué : on a tout d'abord un NAS auquel le client se connecte, le NAS relaie les infos à notre proxy radius, qui les relais lui au radius final qui se charge de l'authentification. Le proxy choisis à quel serveur radius final (ou encore proxy) renvoyer l'info suivant le realm fournis par le supplicant 802.1x du client. Le realm (royaume en français) est le domaine classiquement de l'utilisateur. En ce qui nous concerne on fera donc naturellement la distinction entre les domaines bordeaux1, bordeaux2 etc...

Du coup, il faut désactiver dans le fichier users la partie définition des utilisateurs avec leur pass, et le type d'authentification que nous avions au préalable mis en oeuvre en guise de test.

Et l'on renseigne le fichier proxy.conf de la manière suivante :

```
realm bordeaux1 {
      type            = radius
      authhost        = ip_leaf_radius:1812
      accthost        = ip_leaf_radius:1813
      secret          = textbdx1
      nostrip
}
```

Il existe un module tlm_realm qui décode le userid envoyé par le supplicant client en user@realm. Le proxy peux ensuite faire son choix d'aguillage en fonction de son fichier de configuration.

Côté radius final, on fait actuellement de l'authentification PAP dans un tunnel TTLS, du coup on configure notre radius pour faire du TTLS.

Pour ce qui est du client, il faut bien faire attention lors de la conf de l'identité du client où on mets notre realm, pour en revenir à xsupplicant :

```
# ce que le proxy recoit et découpe en user@realm pour prendre \
# ensuite la décision de proxyfication
identity = <BEGIN_ID>mgeli@bordeaux1<END_ID>

eap_ttls {
    ...
    phase2_type = pap
    pap {
      username = <BEGIN_UNAME>mgeli<END_UNAME>
      password = <BEGIN_PASS>test<END_PASS>
    }
```

Ici le username de pap est celui utilisé par le radius final pour authentifier l'utilisateur, du coup si l'on a un backend LDAP qui nécessite un user@realm, il faudrait remettre

```
<BEGIN_UNAME>mgeli@bordeaux1<END_UNAME>
```

Pour le client Windows SecureW2 il faut donc mettre dans le champ *identity* user@realm, et dans lors du prompt dans *username* mettre user@realm et rien dans domaine.

Attention il risque d'y avoir un problème côté LDAP...

subsectionLDAP + freeradius

La situation est la suivante : j'ai un NAS qui communique avec un proxy radius (oldbxnms) qui renvoit les infos sur le radius de ganmyede (laptop). Je vais donc installer un LDAP sur ganymede :

5.2.7 Install du LDAP

suivre les instructions de la section précédente.

5.2.8 Config du radius

Le serveur tel quel ne trouvera pas rlm_ldap il faut le compiler :

```
 # apt-get install libldap2-dev
 $ cd src/modules/rlm_ldap
 $ ./configure
 $ make
 # make install
```

radiusd.conf

```
ldap {
server = "ldap.u-bordeaux1.fr"
# identity = "cn=admin,o=My Org,c=UA"
# password = mypass
basedn = "ou=people,dc=drimm,dc=u-bordeaux1,dc=fr" # "o=My Org,c=UA"
filter = "(uid=%{Stripped-User-Name:-%{User-Name}})"
# base_filter = "(objectclass=radiusprofile)"

# set this to 'yes' to use TLS encrypted connections
# to the LDAP database by using the StartTLS extended
# operation.
# The StartTLS operation is supposed to be used with normal
# ldap connections instead of using ldaps (port 689) connections
start_tls = no

        # access_attr = "dialupAccess"
access_attr = "wifiAccess"

dictionary_mapping = ${raddbdir}/ldap.attrmap
ldap_connections_number = 5
        password_attribute = userPassword
timeout = 4
timelimit = 3
net_timeout = 1
             }

    authorize {
            ...
            ldap
    }

   authenticate {
            ...
            eap
```

```
   }
\end{verabtim}

ldap.attrmap\\
Ce fichier contient le mapping des nom de variables Radius a faire
correspondre avec celle de LDAP, ce qui nous intéresse ici c'est de
récupérer l'attribut LDAP userPassword et de le stocker dans la
variable RADIUS Password on mets donc :
\begin{verbatim}
 checkItem       Password                          userPassword
```

Precedemment dans la radiusd.conf on a spécifié que l'existence de l'attribut LDAP wifiAccess déterminait le bon déroulement des opérations. Il faut donc l'avoir crée dans la base LDAP pour l'utilisateur en question. Se reporter à ma page LDAP pour des infos sur la création de nouveaux attributs LDAP.

On relance les radius, et on obsèrve la magie opérer...

5.2.9 Freeradius EAP/TLS

On peux aussi si l'on veux s'assurer de l'identité des clients, utiliser EAP-TLS. Dans ce cas, on doit renseigner côté supplicant client en plus du certificat racine, le certificat client. On doit vérifier que le champs `identity` correspond bien à celle du certificat client. Dans mon cas, le certificat client est signé par le serveur nocatauth.u-bordeaux.fr pour le nom d'utilisateur matth. Pour plus de détail sur les commandes OpenSSL pour créer et signer un certificat client voire ma page OpenSSL

```
 allow_types = eap_tls
 identity = <BEGIN_ID>matth<END_ID>
 ...
 eap_tls {
     user_cert      = /etc/xsupplicant/tls/client_cert.pem
     user_key       = /etc/xsupplicant/tls/client_key.pem
     user_key_pass  = <BEGIN_PASS>test<END_PASS>
     root_cert      = /etc/xsupplicant/tls/ca_cert.pem
     crl_dir        = /etc/xsupplicant/tls
     chunk_size     = 1398
     random_file    = /etc/xsupplicant/tls/random

     # To enable TLS session resumption, you need to set the following
     # value to "yes".  By default, session resumption is disabled.
```

```
    # session_resume = yes
}
...
```

Niveau radius, si l'on veut uniquement faire du EAP/TLS on commente de la partie authorize la ligne files ainsi, le fichier users avec notre précédente config n'interférera pas. Il n'y a pas d'autre modification à faire puisqu'on a deja activé EAP/TLS pour TTLS dans eap.conf.

5.2.10 Support

- un paquet de RFC sur radius (http ://www.freeradius.org/rfc/)
- Archive de la ML freeradius-users
- Le canal IRC #freeradius sur irc.freenode.net

5.3 gestion de la PKI

5.3.1 Application

Shibboleth : communication entre les différentes briques, et le client.

Freeradius : chiffrement de l'authentification

5.3.2 Configuration préalable

Pour booster les procédure suivante, il est préférable de renseigner certaines valeurs dans le fichier de configuration openssl.cnf trouvé sous debian dans /etc/ssl

```
countryName_default             = FR
stateOrProvinceName_default     = France
0.organizationName_default      = REAUMUR
```

Pour les certificats en vue d'exportation vers des clients Windows XP, On crée le fichier xpextensions suivant :

```
[ xpclient_ext]
extendedKeyUsage = 1.3.6.1.5.5.7.3.2
[ xpserver_ext ]
extendedKeyUsage = 1.3.6.1.5.5.7.3.1
```

5.3.3 Autorité de certification (CA)

Il existe un script pour faire cela sous Debian à ce jour il se trouve là : /usr/lib/ssl/misc/CA.sh. On le lance du répertoire /etc/ssl, ou d'où on veux tant que ca reste cohérent avec l'option `dir` de /etc/ssl/openssl.cnf On invoque ce script :

```
$ /usr/lib/ssl/misc/CA.sh -newca
```

A partir de là, on devrait avoir dans le répertoire indiqué dans openssl.cnf le fichier de certificat racine, ainsi que sa clef privée.

Remarque : Un certificat est en crypto asymétrique la clef publique.

Nettoyage (en vue d'un export pour WindowsXP)

```
$ openssl x509 -in cacert.pem -out cacert.crt
```

Le but étant lors de la configuration d'un client en mode d'authentification par EAP/TTLS par exemple (qui ne necessite qu'un certificat côté serveur) de placer ce certificat sur le client (disponible classiquement sur une page WEB de l'autorité de certification ou de celle du serveur permettant l'acces. Le client l'importe alors avec son client. C'est un moyen pour le client de ne pas faire du TTLS avec n'importe qui, et donc de donner ses mots de passe en clair à n'importe quelle hacker sui se serait interposé. (oui, le tunnel entre les deux entitées à beau être en TLSv3, l'authentification se faisant par PAP, le mot de passe transite en clair dans le tunnel...)

5.3.4 Certificat serveur

Création

```
# openssl req -new -nodes -keyout server_key.pem \
  -out server_req.pem -days 730 -config openssl.cnf
```

On vient de créer un certificat en attente d'être signé (request) server_req.pem et sa clef privée associée server_key.pem

Signature

Signons-le maintenant avec notre authorité de certification toute fraîche :

```
# openssl ca -config openssl.cnf \
 -policy policy_anything -out server_cert.pem \
 -extensions xpserver_ext -extfile xpextensions \
 -infiles server_req.pem
```

Nettoyage

On enlève toute les lignes avant la balise —BEGIN CERTIFICATE— ca peut se faire à la main, ou automatiquement avec :

```
openssl x509 -in server_cert.pem -out server_cert.crt
```

Combo

On peux concaténer les parties privées et pulique dans un seul fichier et ce sera cet unique fichier que l'on indiquera à freeradius par exemple pour la partie clef, et certificat. (peu d'interêt selon moi)

```
# cat server_key.pem server.cert.pem > server_keycert.pem
```

5.3.5 Création d'un certificat client

Problème

suite à l'erreur :

```
client:invalid type in 'policy' configuration
```

j'ai modifié dans /etc/ssl/openssl.cnf dans la section [policy_anything]

```
commonName = optional
```

Création

```
# openssl req -new -keyout client_key.pem \
 -out client_req.pem -days 730 -config openssl.cnf
```

Signature

```
# openssl ca -config openssl.cnf \
 -policy policy_anything -out client_cert.pem \
 -extensions xpclient_ext -extfile xpextensions \
 -infiles client_req.pem
```

Windows XP

On exporte au format PKCS12 le certificat client :

```
# openssl pkcs12 -export -in client_cert.pem \
  -inkey client_key.pem -out client_cert.p12 -clcerts
```

5.3.6 Références

- http ://www.linuxjournal.com/article/8095
- http ://www.alphacore.net/spip/article.php3 ?id_article=45

5.4 Traffic shaping

```
#!/bin/bash

######
##
#   Script example to shape incoming traffic
#   knowing the dst IP. (restrict download from
#   the client point of view)
#
#   require:  o tc (iproute2 package)
#             o QOS Networking options in Linux kernel
#               (u32, cbq/htb and sfq mainly)
#
# ressources : http://www.lartc.org/howto/ (EN)
#              http://www.linux-france.org/prj/inetdoc/guides/lartc/ (FR)
#

# Revision 1.0 27/08/2005 mgeli
# - code cleanup from blino's advices, added show func
#
# Revision 0.2 26/08/2005 mgeli
# - modified (handle mainly) to be called from a Perl CGI
#
# Revision 0.1 25/08/2005 mgeli
# - draft script ugly/buggy



BANDWIDTH=10mbit
ACTION=$1
shift

usage() {
    cat <<EOF
```

```
    ./bandwith_limit init if
    ./bandwith_limit add if ip in_bandwith class
    ./bandwith_limit del if ip class
    ./bandwith_limit show if
    if : interface
    ip : IP, or CIDR notation
    in_bandwith : bandwith in KB/s
    class : number of the leaf class
    'show' is used to see the handle number to delete a shaper
    NB: you have to 'init' an interface before
    shaping any client
EOF
}

shaper_add() {
    DEV=$1
    FILTERMASK=$2
    IN_BANDWIDTH=$(($3*8))
    CLASS=$4
    HANDLE=$CLASS

    # HTB version
    tc class add dev $DEV parent 10:1 classid 10:$CLASS htb  \
    rate ${IN_BANDWIDTH}kbit burst 15k
    tc qdisc add dev $DEV parent 10:$CLASS handle ${HANDLE}: \
    sfq perturb 10
    tc filter add dev $DEV parent 10:0 handle 800::$CLASS    \
    protocol ip prio 100 u32 match ip dst $FILTERMASK flowid 10:$CLASS

}

shaper_init() {
    DEV=$1

    # HTB version : more accurate
    tc qdisc add dev $DEV root handle 10: htb default 30
    tc class add dev $DEV parent 10: classid 10:1 htb rate   \
    $BANDWIDTH burst 15k
}

shaper_del() {
```

```
    DEV=$1
    FILTERMASK=$2
    HANDLE=$3
    CLASS=$HANDLE

    # HTB powah version
    tc filter del dev $DEV parent 10:0 handle 800::$CLASS    \
    protocol ip prio 100 u32 match ip dst $FILTERMASK flowid 10:$CLASS
    tc qdisc del dev $DEV parent 10:$CLASS handle ${HANDLE}: sfq perturb 10
    tc class del dev $DEV parent 10:1 classid 10:$CLASS
}

shaper_show() {
    DEV=$1
    tc filter show dev $DEV
}

case $ACTION in
    "init")
if [ "$1" != "" ]; then
    shaper_init $1
else
    usage
fi
;;
    "add")
if [ $# -eq 4 ]; then
   shaper_add $*

else
    usage
fi
;;
    "del")
if [ $# -eq 3 ]; then
    shaper_del $*
else
    usage
fi
;;
    "show")
```

```
if [ $# -eq 1 ]; then
    shaper_show $1
else
    usage
fi
;;
    *)
usage
;;
esac


#
## CBQ relics
#

# method skipped not accurate enough when users were having
# bandwith utilisation too inegal.

# add

# tc class add dev $DEV parent 10:1 classid 10:$CLASS    \
# cbq bandwidth $BANDWIDTH rate ${IN_BANDWIDTH}kbit      \
# allot 1514 weight 85kbit prio 5 maxburst 20 avpkt 1000 bounded
# tc qdisc add dev $DEV parent 10:$CLASS sfq quantum 1514b perturb 15
# tc filter add dev $DEV parent 10:0 handle 800::$HANDLE \
# protocol ip prio 100 u32 match ip dst $FILTERMASK flowid 10:$CLASS

# del

# tc filter del dev $DEV parent 10: protocol ip prio 100 \
# handle 800::$HANDLE u32
# tc qdisc del dev $DEV parent 10:$CLASS
# tc class del dev $DEV parent 10:1 classid 10:$CLASS


# init

# tc qdisc add dev $DEV root handle 10: cbq bandwidth $BANDWIDTH avpkt 1000
# tc class add dev $DEV parent 10:0 classid 10:1 cbq bandwidth  \
# $BANDWIDTH rate $MAX_LINK_RATE allot 1514 weight 1Mbit prio 8 \
```

```
# maxburst 20 avpkt 1000
```

Chapitre 6

Glossaire

802.1x : Standard de l'IEEE[1] pour un contrôle d'acces au réseau par port. Il fournit un moyen d'authentifier le matériel attaché à un port sur un LAN[2]. Le protocole de tunneling utilisé pour communiquer avec l'entité de contrôle est une extension de PPP nommé EAP.

802.11i : Spécification de mécanismes de sécurité pour renforcer la norme 802.11. L'implémentation de la Wifi-Alliance de 802.11i se nomme WPA2.

WPA : Wi-Fi **P**rotected **A**ccess, en tant que solution intermédiaire pour pallier aux faiblesses de WEP. Il s'agit d'une implémentation partielle par la Wifi-Alliance de la norme 802.11i.

Certificat X.509 : Le certificat est la pièce qui associe à une entité (client ou serveur), sa clé publique. La certification est un élément fondamental du système de cryptographie à clé publique.

PPP : le **P**rotocole **P**oint à **P**oint est un protocole décrit par le RFC 1661, fortement basé sur HDLC[3], qui permet d'établir une connexion de type réseau entre deux hôtes sur une liaison point à point. Il fait partie de la Couche de liaison (couche 2) du modèle OSI [4].

PAP : Password **A**uthentication **P**rotocol est un protocole d'authentification pour PPP. Il a comme avantage d'être extrêmement simple à implémenter, ce qui permet de l'utiliser pour des systèmes embarqués très légers. Il a cependant le gros défaut de faire circuler le mot de passe d'authentification en clair.

CHAP : Challenge **H**andshake **A**uthentication **P**rotocol est un protocole d'authentification pour PPP à base de challenge, ce qui le rend bien plus

[1] Institute of Electrical and Electronics Engineers
[2] Local Area Network
[3] High Level Data Link Control
[4] Open Systems Interconnection

discret que son pendant PAP : il est très dur pour un espion d'obtenir le mot de passe en écoutant simplement la conversation. Ce protocole est défini dans la RFC 1994.

EAP : **E**xtended **A**uthentication **P**rotocol est une extension de PPP qui permet de rajouter différents moyens d'authentification.

HTTP : **H**yper **T**ext **T**ransfert **P**rotocol, utilisé comme moyen de communication entre les client et serveur web pour la visualisation de contenu.

HTTPS : version sécurisé de HTTP utilisant SSL dans lequel le serveur est authentifié, le client non.

TLS : **T**ransport **L**ayer **S**ecurity communément nommé SSLv3.1 est une évolution de SSLv3. Il se place au dessus d'un protocole de transport tel que TCP, ou encore EAP et est basé sur la cryptographie asymétrique.

EAP/TLS : Considéré comme un des moyens les plus sûr pour s'authentifier sur un LAN (filaire ou non) il consiste en l'encapsulation d'une couche TLS au dessus de EAP pour discuter avec le système authentifiant. Il est necessaire d'avoir un certificat côté client, ce qui en fait son talon d'achille du point de vue du deploiement. Il béneficie d'une large adoption quelque soit le système utilisé.

EAP/TTLS : EAP **T**unneled **T**ransport **L**ayer **S**ecurity est un standard toujours à l'état de brouillon qui est néammoins supporté par de nombreux matériels. Sa différence avec EAP/TLS est qu'il utilise uniquement un certificat côté serveur.

EAP/MD5 : Système d'authentification basé sur le hashage du mot de passe client. Sécurité faible si les transactions peuvent être écoutées : attaque par dictionnaire.

PEAP : Protocole remplissant la même fonction que EAP/TTLS, proposé par Cisco et Microsoft. Ils différent par le fait qu'ils ne nécessitent pas de client (supplicant) externes sous le système Windows avec un Service Pack récent.

NAS : **N**etwork **A**ccess **S**erver, surveille l'acces à une ressource (acces réseau par exemple) et fait office d'intermédiaire entre le client, et le serveur d'acces dont dépends la décision d'acces.

Iptables : Firewall IP developpé par l'équipe Netfilter pour GNU/Linux.

NAT : translation d'adresse réseau. Procédé permettant à une routeur de relayer des paquets pour d'autres machines en modifiant les champs IP source ou destination.

DNAT : NAT de destination, modifie le champs IP de destination d'un paquet transitant par le routeur.

SNAT : NAT source, modification du champs IP source du paquet transitant par le routeur.

Masquerading : SNAT spécifique ou l'adresse sources est fixée par l'adresse de l'inteface émetrice du routeur.

VLAN : **V**irtual **L**ocal **A**rea **N**etwork, réseau local regroupant un ensemble de machines de façon logique et non physique.

SSO : **S**ingle **S**ign **O**n, procédé d'authentification unique révelant des moyens mis en oeuvre pour fédérer l'authentificaton entre des partenaires pour un accès à leurs ressources.

Tunneling : Principe consistant à court-circuiter l'enchaînement standard des différents protocoles réseau du modèle OSI. Par exemple réinjecter au niveau de la couche de transport (TCP,UDP...) une sous-partie du modèle OSI débutant au niveau 2 (transmission de données) jusqu'au niveau applicatif.

Radius : **R**emote **A**ccess **D**ial **I**n **U**ser **S**ervice, un serveur qui permet d'authentifier et faire de la gestion de comptes utilisateur (accounting). Très utilisé par les fournisseurs d'accès à internet, il peux être étendu pour tout réseau qui necessite une authentification et la gestion de comptes centralisé.

GNU : Le projet GNU a été lancé par Richard Stallman en 1984, alors qu'il travaillait au laboratoire d'intelligence artificielle du MIT, afin de créer un système d'exploitation libre et complet et, d'après ses mots,

> *ramener l'esprit de coopération qui prévalait dans la communauté informatique dans les jours anciens*

GNU/Linux : Système d'exploitation construit autour d'un noyau developpé par une communauté de programmeurs à travers internet et d'un envrionnement de travail crée par le projet GNU ayant de nombreux points commun avec les systèmes UNIX proriétaires.

Index