YaCaP – Yet another Captive Portal
Un portail captif mutualisé
pour les Universités et établissements lorrains

Alexandre SIMON
CIRIL - Centre Interuniversitaire de Ressources Informatiques de Lorraine
Rue du Doyen Roubault
54500 VANDOEUVRE-LÈS-NANCY - FRANCE
Alexandre.Simon@ciril.fr

Loïc BARREAU
CIRIL - Centre Interuniversitaire de Ressources Informatiques de Lorraine
Rue du Doyen Roubault
54500 VANDOEUVRE-LÈS-NANCY - FRANCE
Loic.Barreau@ciril.fr

Sébastien MOROSI
CIRIL - Centre Interuniversitaire de Ressources Informatiques de Lorraine
Rue du Doyen Roubault
54500 VANDOEUVRE-LÈS-NANCY - FRANCE
Sebastien.Morosi@ciril.fr

Résumé

YaCaP est un portail captif développé par le CIRIL pour répondre à la montée en charge des réseaux wifi et aux exigences d'authentification de ces réseaux « ouverts ». YaCaP propose une architecture originale basée sur une infrastructure centralisée et mutualisée de serveurs et de routeurs matériels. L'infrastructure s'intègre facilement aux réseaux et aux systèmes d'information déjà existants et ne requiert qu'un minimum de prérequis sur les postes clients.

Le fonctionnement côté client est assez classique en comparaison avec les autres solutions de portail captif. La spécificité de YaCaP est sa capacité à fédérer sur une même plate-forme un ensemble de populations différentes s'authentifiant sur les systèmes d'information de leurs établissements respectifs. Tous les paramètres et les phases de connexion sont configurables et surchargeables (par les administrateurs des SI) en fonction du réseau d'établissement hôte.

La solution a été développée pour proposer un maximum de souplesse et des niveaux élevés de performances, de robustesse et de haute disponibilité pour un service optimum. Le développement a tenu compte du passage à l'échelle en adoptant des solutions adaptées à chaque fonction et en autorisant une implémentation avec différents niveaux de services.

Mots clefs

portail captif, accès authentifiés, *wifi*, fédération de systèmes d'information, haute disponibilité, sécurité des réseaux ouverts.

1 Historique, besoins et choix

YaCaP (*Yet another Captive Portal*) est un portail captif développé par le CIRIL depuis février 2005. Ce portail a pour but de proposer un accès réseau contrôlé après authentification de l'utilisateur sur le système d'information de son établissement.

Le projet YaCaP a commencé début 2005 suite à l'émergence des déploiements de bornes *wifi* qui nécessitaient un minimum d'authentification pour l'accès au réseau. A cette époque, des solutions de portails captifs existaient (*m0n0wall*, *chillispot*, *NoCatAuth...)* mais suscitaient toutes les mêmes interrogations pour une mise en production.

Ces solutions dites « en rupture », s'intercalant entre le client et l'*Internet*, posent des problèmes d'architecture et de déploiement : comment déployer autant de portails captifs que de sites et de réseaux captifs ? Comment s'intégrer aux réseaux existants sans avoir à repenser toute la topologie autour de ce service de portail captif ?

Ces solutions, s'appuyant sur des technologies de filtrage et de routage logicielles (c'est le portail captif en rupture qui autorise et route les flux *IP*), rendent difficile l'évaluation de la montée en charge et le provisionnement du nombre de serveurs à déployer.

Les mécanismes d'authentification et d'autorisation de ces solutions sont relativement « basiques ». L'interopérabilité avec les systèmes d'information des établissements n'est pas assez poussée, voire incomplète (ex. pas de *CAS*). Elle ne permet pas une granularité suffisante des accès (ex. type d'autorisation, horaires d'accès... selon la population).

L'exportation des journaux d'activités, un point incontournable et sensible, n'est pas toujours prise totalement en compte par ces solutions. En effet, elles ne

permettent pas d'exporter facilement les *logs* d'authentification et d'accès, ni même d'établir des croisements entre authentification et usage.

Enfin, ces solutions ne proposent pas systématiquement de haute disponibilité native, ni de facilité d'extension pour faire face à une montée en charge. Comment faire dans ce cas, sans complexifier l'infrastructure, pour proposer un service hautement disponible, dont les performances ne sont pas limitées au bout de quelques semaines ou quelques mois ?

Ces solutions, plutôt adaptées à de « petits sites », ne sont pas déployables à grande échelle sur un réseau régional multi-établissement avec une logique de mutualisation forte. Le développement de YaCaP avait donc pour but de pallier à ces constatations, mais également d'apporter des solutions à d'autres problèmes non pris en compte.

2 Architecture

2.1 Principe général

YaCaP est constitué de deux infrastructures distinctes : une infrastructure système (les serveurs captifs) qui interagissent sur l'infrastructure réseau (les routeurs captifs). L'idée est de continuer à utiliser des routeurs matériels pour autoriser et router les flux *IP*, les serveurs de l'application YaCaP configurant « à la volée » ces matériels en fonction des connexions et déconnexions des utilisateurs. Ce principe de fonctionnement est illustré par la Figure 1. Pour simplifier le schéma, il n'apparaît qu'un routeur et un serveur captifs mais pour des raisons de performance et de haute disponibilité ces matériels peuvent être dupliqués.

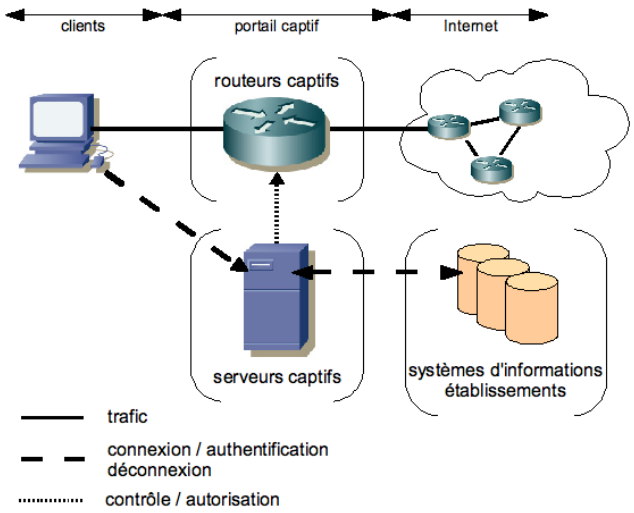

Figure 1: Architecture générale

L'originalité de cette architecture permet toutes les implémentations souhaitées et résout un certain nombre de problèmes. Il n'y a plus de solution logicielle en rupture, ce sont des matériels réseaux qui prennent en compte les flux, la montée en charge est donc connue et maîtrisée. Les routeurs ne sont pas limités, ni en modèle (ou marque), ni en nombre, ni en localisation, ce qui rend possible la création d'un réseau captif (géré en central) sur un réseau routé à l'autre bout de l'*Internet*. Le type de réseau captif n'est pas non plus limité car YaCaP prend en compte le niveau 3 et peut donc s'adapter à des réseaux filaires ou *wifi*.

L'application YaCaP peut être répartie sur un ensemble de serveurs, permettant d'isoler les fonctions, de les doubler et donc de rendre l'application hautement disponible.

YaCaP s'appuie, pour les phases d'authentification et d'autorisation des utilisateurs, sur les systèmes d'information des établissements souhaitant déployer des réseaux captifs. Cette interface se voulant « ouverte », YaCaP dispose de plusieurs méthodes de communication (*LDAP*, *Radius* et *CAS*) vers les SI. Il tente d'être générique en s'adaptant aux contraintes de la base d'information distante (schéma, arborescence, attributs...).

2.2 Mutualisation centralisée et sites distants

Les avantages de la mutualisation et de la centralisation d'applications telles que YaCaP ne sont plus à démontrer. Malheureusement, les réseaux mis en oeuvre ne permettent pas toujours un déploiement total de ce type d'application et c'est pourquoi YaCaP a voulu s'adapter à cette situation en prenant en compte un maximum de topologies différentes. Les réseaux rencontrés sont souvent de deux types :

- ✔ Les réseaux commutés de niveau 2 autorisent la création et le transport de *vlans* (*LAN* ou *MAN*), il est dans ce cas facile d'interconnecter n'importe quel *vlan* du réseau sur l'infrastructure centralisée.
- ✔ Les réseaux routés de niveau 3 n'autorisent pas le transport de *vlans* du fait de l'utilisation de routage « classique ». Il est néanmoins possible de rendre « transparent » le réseau routé traversé par l'utilisation de tunnels et ainsi d'interconnecter directement le réseau distant à l'infrastructure centralisée.

La Figure 2 illustre ce principe de mutualisation centralisée des ressources (*ie*. les routeurs et serveurs captifs) ainsi que le support de réseaux clients *MAN/LAN* commutés et *WAN* routés.

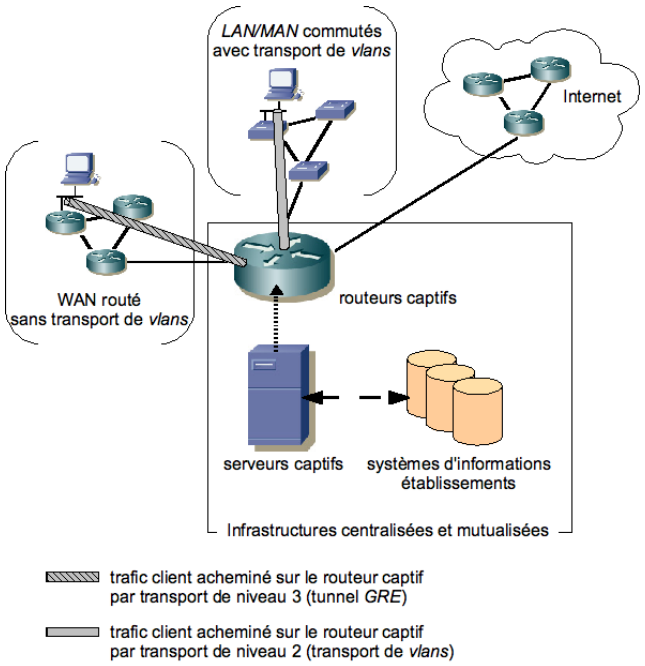

Figure 2: Centralisation et sites disants

Le portail captif centralisé en un point favorise la mutualisation des ressources et facilite l'exploitation de l'infrastructure en comparaison avec une solution de type « boîte noire » sur chaque site.

3 Fonctionnement

3.1 Prérequis réseaux et postes clients

La solution YaCaP tente d'éliminer les contraintes côté réseau et côté client en utilisant des mécanismes standards et largement utilisés. Tous les mécanismes « propriétaires » ou trop contraignants pour un déploiement ont donc été éliminés.

Le réseau physique ne nécessite aucun déploiement de protocoles particuliers pour le bon déroulement d'une session utilisateur. Le poste client ne doit suivre qu'un minimum de contraintes : une pile *TCP/IP* avec *DHCP* et un navigateur avec le support des *popups*, des *cookies* et du langage *javascript*.

Le protocole *HTTPS* garantit la sécurité des échanges entre le client et le portail captif. Tout autre mécanisme de chiffrement sur le medium n'est pas à la charge du portail mais peut être mis en place sur l'infrastructure réseau du site (*WEP*, *WPA*, *802.1X*) ou à l'initiative de l'utilisateur une fois connecté (*VPN* vers intranet, *HTTPS* vers sites sensibles...).

3.2 Configuration automatique du poste client

Un serveur *DHCP* central distribue les paramètres réseaux nécessaires à la configuration du poste client. Ces informations dépendent du réseau hôte et sont renseignées à la création du *vlan* captif correspondant. Ainsi, deux postes clients sur deux *vlans* d'un même établissement peuvent se voire attribuer des configurations distinctes : serveurs *DNS*, nom de domaine...

L'utilisation de ce service n'est pas obligatoire mais permet, en plus d'une facilité d'exploitation pour l'administrateur du réseau, la collecte et l'ajout des adresses *MAC* dans les journaux d'activités facilitant la recherche de poste client ou le suivi des utilisateurs.

3.3 Déroutage du premier trafic *web*

Les flux *HTTP* (*80/tcp*) issus du poste client sont redirigés par le routeur vers l'infrastructure serveur YaCaP et en particulier vers l'*URL* du portail captif propre à l'établissement du *vlan* hôte. L'infrastructure portail captif « héberge » en quelque sorte tous les sites de portails captifs des établissements souhaitant utiliser YaCaP. L'intérêt est qu'en fonction du *vlan* source, l'utilisateur arrive sur les « bonnes » pages d'informations et d'authentification de l'établissement concerné.

Il choisit alors son scénario d'authentification en fonction de sa qualité (étudiant, personnel de l'université, invité), et/ou de son établissement d'origine. Cette phase de redirection est illustrée par la Figure 3.

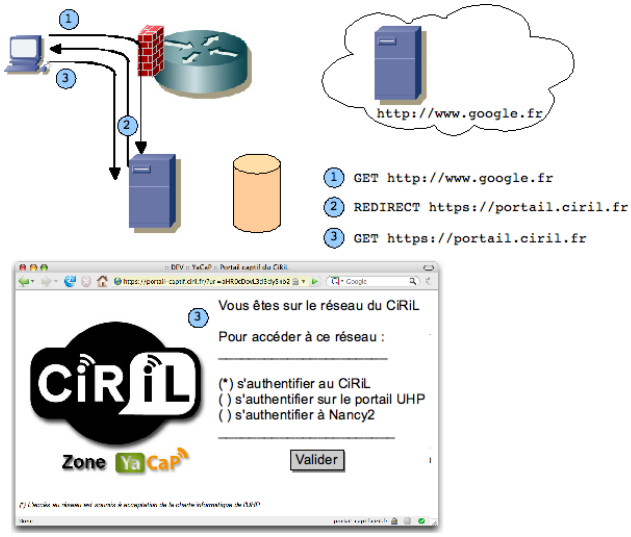

Figure 3: Phase de redirection

Ces pages sont totalement modifiables (contenu et forme *HTML)* par les personnes autorisées à les publier pour un établissement donné. Le portail captif ajoute « à la volée » dans le corps de la page la liste des scenarii (liste de choix où l'utilisateur va pouvoir s'authentifier) en fonction du *vlan* hôte.

3.4 Authentification, autorisation et utilisation

Le choix du scénario conditionne le système d'information à interroger pour l'authentification et l'autorisation. La liste des scenarii disponibles sur un *vlan* captif est renseignée à la demande de l'établissement dans la base de configuration de YaCaP.

Une fois l'utilisateur dûment authentifié par son SI et autorisé en fonction d'attributs collectés par YaCaP (compte autorisé à se connecter, horaires de validité...), le serveur captif programme « à la volée » le routeur en positionnant :

- ✔ les *ACLs* nécessaires à l'autorisation du trafic pour l'adresse *IP* attribuée
- ✔ et la configuration évitant la redirection initiale.

Un *popup HTML* muni d'un *cookie* de session est ouvert sur le poste client et son navigateur est redirigé vers l'*URL* demandée initialement ou inconditionnellement vers une autre *URL* paramétrable (l'ENT de l'établissement par exemple). Cette phase de connexion est présentée par la Figure 4.

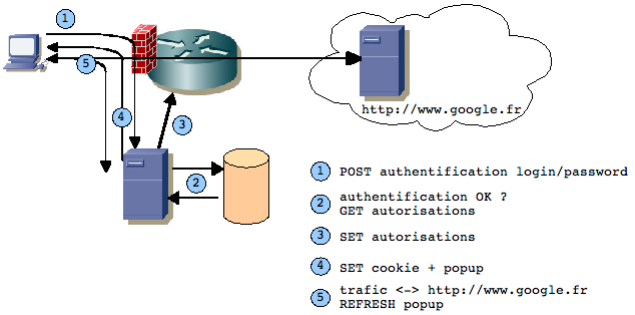

Figure 4: Phase d'authentification, d'autorisation et d'utilisation

Le contenu du popup est rafraîchi à intervalle régulier attestant de la présence et de l'activité de l'utilisateur. A chaque rafraîchissement, le *cookie* de session est rejoué sur le portail captif et sa validité est vérifiée : le *cookie* présenté est-il celui généré initialement pour ce poste ? (test d'usurpation), le *cookie* présenté correspond-il à une session en cours ? La présence du *popup* lors de la phase d'utilisation normale est illustrée par la Figure 5.

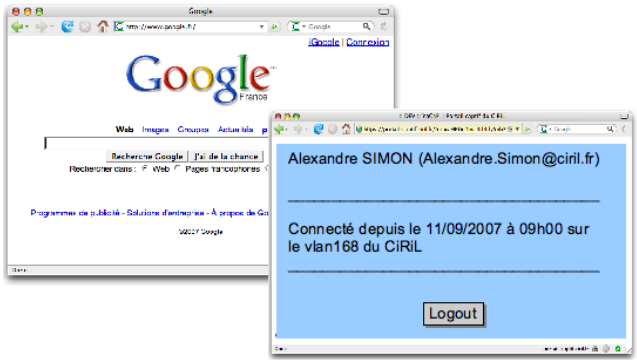

Figure 5: Phase d'utilisation et popup

Au moindre « défaut » de ce *cookie* de session, l'utilisateur est automatiquement déconnecté et doit donc repasser par la phase de redirection initiale pour se réauthentifier.

Les sessions en cours et leur historique sont stockés dans la base de données YaCaP et tracés dans les journaux d'activités.

3.5 Systèmes d'information et attributs utilisés

YaCaP est prévu pour supporter les authentifications *CAS*, *LDAP / Active Directory*, *Shibboleth* et *Radius*. A ce jour seuls *CAS*, *LDAP/AD* et partiellement *Radius* sont implémentés. La Figure 6 présente les pages d'authentification *LDAP* et *CAS*.

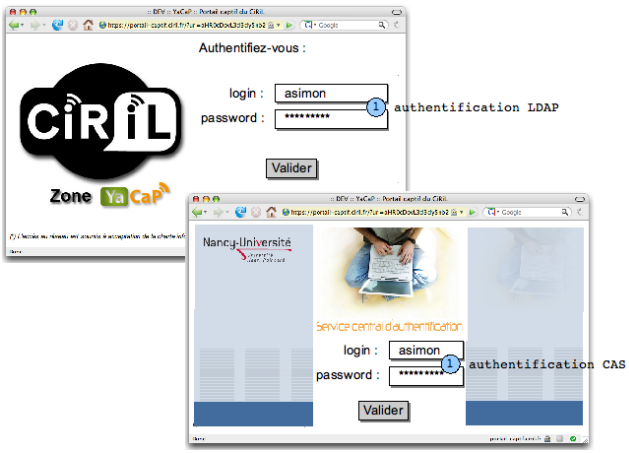

Figure 6: Authentification LDAP et CAS

Les paramètres de configuration d'un *vlan* pour le fonctionnement de YaCaP sont renseignés à sa création dans une base de données : *ACLs* à positionner, horaires d'accès, *timeouts* de connexion, *URL* de redirection...

Ces attributs peuvent être présents dans les systèmes d'information des établissements interrogés et permettent la surcharge des attributs par défaut. YaCaP utilise *LDAP* ou *Radius* pour consulter ces paramètres. Un schéma d'extension des systèmes d'information est proposé aux établissements souhaitant utiliser cette notion de surcharge. Les administrateurs ont donc l'autonomie et la maîtrise de la politique de sécurité de leurs *vlans*.

Après authentification, les utilisateurs peuvent se voir attribuer des autorisations d'accès différentes en fonction d'informations spécifiques à leur fiche dans le SI de leur établissement. Bien que chaque *vlan* ait une configuration par défaut, définie à sa création en fonction de la politique de sécurité adoptée par l'établissement, la granularité de configuration la plus fine est donc l'utilisateur. Un même *vlan* peut ainsi accueillir deux utilisateurs ayant un même « profil » YaCaP mais ayant des droits réseau différents du fait de la surcharge d'attributs dans leurs fiches utilisateurs respectives : *ACLs* positionnées, *timeouts* différents...

YaCaP n'a pas connaissance des utilisateurs mais seulement des systèmes d'information laissant une entière liberté de gestion aux établissements.

YaCaP permet également la mutualisation inter-établissement en autorisant une authentification transversale sur tous les systèmes d'information configurés. Par exemple, un *vlan* de bibliothèque universitaire pourra accueillir l'ensemble des étudiants des universités lorraines, chaque étudiant s'authentifiant dans son établissement d'origine.

3.6 Déconnexion

Une session utilisateur prend fin par :

- ✔ la déconnexion explicite par l'utilisateur (bouton de *logout* dans le *popup*) ;
- ✔ la déconnexion de l'utilisateur par l'administrateur du *vlan* hôte via l'interface d'administration ;
- ✔ la détection par le portail YaCaP de l'inactivité du *popup* (*soft-timeout*) : l'utilisateur a quitté son navigateur ou fermé le *popup*, le poste client s'est déconnecté du réseau (portable fermé) ;
- ✔ une limite inconditionnelle qui déconnecte tout utilisateur, actif ou non, au delà d'une certaine durée de connexion (*hard-timeout*), pour éviter les connexion « infinies » ;
- ✔ un défaut de vérification du *cookie* de session utilisateur (tentative d'usurpation...).

Ces deux limites (*soft* et *hard timeout*) sont renseignées à la création du *vlan* captif et sont donc fonction du réseau hôte. La phase de déconnexion est illustrée par la Figure 7.

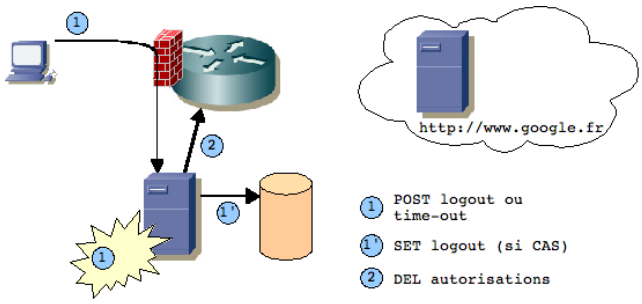

Figure 7: Phase de déconnexion

Pour toute déconnexion, le routeur est re-programmé en réinitialisant les *ACLs* pour ce poste dans la configuration de départ (avant authentification). Le mécanisme de redirection initiale est de nouveau activé.

Les informations de connexion sont mises à jour dans la base de données.

3.7 Interface d'administration

YaCaP offre une interface *web* aux administrateurs des établissements leur permettant de visualiser les configurations et l'activité des utilisateurs sous forme de graphiques, d'historiques et de compteurs.

Les journaux d'activités (connexions et déconnexions avec leurs causes) sont personnalisés (ajout des adresses *MAC*, adresse *IP*, *login* et scénario utilisé), stockés localement et exportés en *syslog* vers chaque établissement.

Les pages *HTML* (pages d'authentification, de déconnexion, *popup*...) sont entièrement libres de forme, le portail renseignant « à la volée » les informations connues de lui seul (liste des scenarii d'authentification) ou les informations dynamiques propres à la session en cours (*login* par exemple) par un système de *template* et de mots clefs. La publication, par *FTP*, de ces pages sur l'infrastructure serveur YaCaP n'est possible que pour les administrateurs de ce service de publication.

3.8 Synthèse des fonctionnalités

En résumé, voici la liste synthétique des fonctionnalités de YaCaP :

- ✔ facilité d'intégration dans un réseau existant
- ✔ indépendance vis à vis de la couche physique et des couches supérieures
- ✔ minimum de prérequis sur le poste client, zéro configuration
- ✔ robuste et extensible (de 1 à plusieurs milliers d'utilisateurs)
- ✔ utilisation de routeurs matériels
- ✔ multi établissement :
  - ○ multi-système d'information
  - ○ multi-scenarii d'authentification
  - ○ multi-protocole d'authentification
  - ○ *templates HTML*
- ✔ mutualisation des ressources et des infrastructures

4 Architecture logicielle et implémentation

L'infrastructure YaCaP devenant le point de sortie obligé pour les *vlans* captifs, une attention particulière a été apportée pour rendre le service performant, robuste et hautement disponible.

Lors du développement et de la mise au point, plusieurs « briques fonctionnelles » ont été identifiées et des efforts tout particuliers ont été mis en oeuvre pour optimiser ces fonctions (performance et robustesse) et pour proposer des possibilités de redondance (haute disponibilité). L'ensemble de ces briques est schématisé sur la Figure 8.

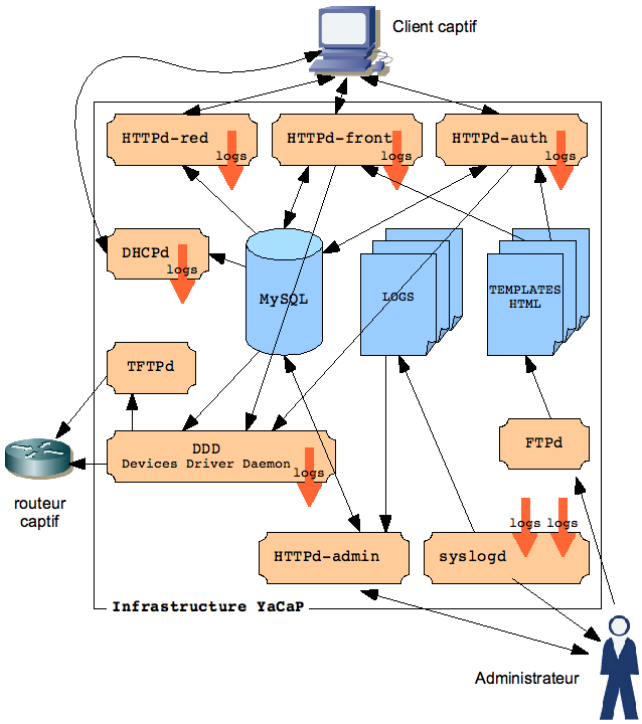

Figure 8: Infrastructure et briques fonctionnelles

4.1 Architecture logicielle

YaCaP a été développé en *Perl* et s'appuie sur un ensemble d'outils libres (se référer à l'Annexe pour la liste et les références des outils utilisés). Ce paragraphe se concentre sur les points techniques les plus importants et explicite les principes mis en oeuvre.

Chaque fois que cela est possible et nécessaire, les informations utiles à la configuration de chaque fonction sont « pré-chargées » au démarrage à partir de la base de données, réduisant à son nombre minimum les interrogations *SQL* lors du fonctionnement. A titre d'exemple les *templates HTML*, la configuration de chaque *vlan*, la liste des scenarii disponibles, les *URLs* de redirection sont pré-chargés dans les différents serveurs *web apache* rendant directement disponibles ces informations aux processus qui servent les requêtes des clients.

Plusieurs serveurs *web apache* sont utilisés pour rendre les services de redirection, d'authentification et de *popup*. Les sollicitations et les ressources consommées sont différentes pour chacun de ces services, justifiant la spécialisation de trois serveurs *apache+mod_perl*. Pour chacun de ces différents serveurs, la compilation et la configuration sont optimisées (*ie.* ne compiler et ne configurer que le strict nécessaire) et le paramétrage est affiné (*ie.* nombre de fils maximum, nombre de requêtes simultanées maximum...). Ces spécialisations garantissent une utilisation rationnelle des ressources pour un service optimum.

La fonction de redirection est en rupture sur tous les flux *HTTP* provenant des postes clients non encore authentifiés. Ce flux correspond « normalement » à l'ouverture du navigateur par le client et au chargement de sa page par défaut. Mais ce flux correspond également à de nombreux autres trafics « licites » ou non : mises à jour automatiques, virus/vers, « autres » logiciels... Avant authentification, tout flux *HTTP* sollicite donc la partie redirection des serveurs captifs. Après expérience, cette sollicitation peut s'avérer très importante (plusieurs milliers de requêtes par seconde) et nécessite l'utilisation d'un *apache+mod_perl* spécialisé (comme expliqué ci-dessus) épaulé par un principe de *blacklist* sur *user-agent* pour limiter les accès aux navigateurs *web* uniquement.

La programmation « à la volée » des routeurs captifs fait l'objet d'un développement spécifique : le démon *DDD (Devices Driver Daemon*). Il est développé selon le modèle client/serveur pour l'extraire de toute autre brique fonctionnelle et le rendre autonome; Les clients sont alors les fonctions et les serveurs ayant besoin d'effectuer les opérations de connexions et de déconnexions des clients. L'utilisation de *threads*, le pré-chargement des configurations et la parallélisation du « pilotage » des routeurs réduisent le temps d'exécution pour ces opérations. D'autre part, une gestion « intelligente » des *access-lists* générées permet de réduire les échanges entre le serveur *TFTP* et les routeurs, d'en limiter la charge et d'optimiser le temps de reconfiguration.

4.2 Implémentation

Comme illustré Figure 8, un certain nombre de « briques logicielles » indépendantes est identifié. La communication et l'interaction entre ces briques sont conçues pour que ces instances puissent être hébergées sur des serveurs différents tout en offrant la possibilité de les redonder avec possibilité d'équilibrage de charge. En particulier, tous les serveurs *HTTP* peuvent être dupliqués en utilisant des solutions et des implémentations libres de *failover* et de *load-balancing* (*Keepalived* et *IPVS*).

Les serveurs *HTTP* supportant le trafic lié au rafraîchissement des *popups* (*HTTPd-front*) peuvent être séparés des autres serveurs *apache* : en cas de souci majeur sur les fonctions de redirection ou d'authentification *(flood* par exemple), les utilisateurs connectés ne sont pas impactés.

La problématique d'accès aux informations de configuration, aux données et aux journaux d'activité, dans le cas d'une répartition des briques sur des serveurs différents est traitée par l'utilisation d'une base de données centrale, la distribution « intelligente » des configurations par l'outil *Cfengine* et la collecte centralisée des traces via *syslog*. La solution de mise à disposition de ces informations à partir d'un système de fichiers partagés (*NFS*, *CIFS*...) a été écartée essentiellement pour des problèmes de performances, de passage à l'échelle et de robustesse.

Le serveur de base de données est au coeur du portail captif. Il contient la configuration de tous les *vlans*, les paramètres et attributs propres à chaque fonction et maintient la table des sessions actives. Ce service peut être hébergé sur deux serveurs physiques, un système de réplication est utilisé pour maintenir la cohésion des tables.

Les possibilités de routage avancé (*source routing*...) du système *Linux* ont été utilisées pour réaliser la configuration réseau des serveurs captifs et pour permettre la segmentation des fonctions (*DHCPd*, redirection initiale, *DDD/TFTPd*...) sur des *subnets IP* différents. Cette segmentation est intéressante notamment pour numéroter « en privé » les services qui ne doivent être accessibles que depuis les clients captifs, pour garantir « l'étanchéité » des échanges et pouvoir désactiver « brutalement » (*shutdown* de l'interface de routage, règle *iptables*...) une fonction en cas de problème de sécurité.

Le routage des *subnets* propres aux serveurs captifs peut être sécurisé par l'utilisation de protocoles tels que *VRRP* ou *HSRP*. Ainsi, dans le cas d'une topologie comptant plus d'un routeur captif, les *subnets* de l'infrastructure serveurs sont routés en plusieurs points garantissant la disponibilité du service en cas d'arrêt d'un élément réseau.

Enfin, une supervision des serveurs, des services et des journaux d'activités permet de remonter automatiquement des alertes en cas de pannes « franches » ou de dépassements de seuils avec une dégradation du service.

5 Les « à côtés » sécurité

L'architecture centralisée de YaCaP et ses facilités d'intégration au sein de réseaux déjà existants permettent d'envisager l'ajout de services complémentaires pour répondre aux problématiques de sécurité des réseaux « ouverts ».

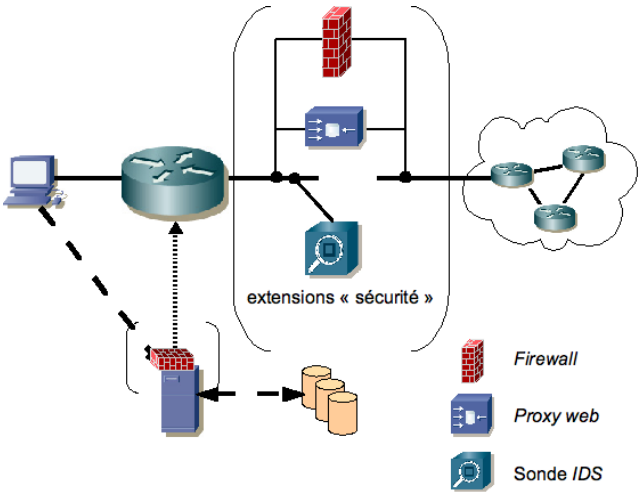

Figure 9: Possibilités d'extensions sécurité

La Figure 9 illustre quelques exemples d'extensions sécurité insérées entre l'infrastructure centralisée et l'*Internet*, point de passage obligé pour l'ensemble du trafic *IP* des clients captifs.

Il est ainsi possible d'insérer un *proxy HTTP* transparent avec l'activation des mécanismes disponibles sur ce type de service tels qu'un antivirus en entrée et/ou en sortie, le filtrage d'*URLs* ou encore la vérification de la « bonne » utilisation du protocole *HTTP* (*ie.* interdire l'encapsulation de trafic non contrôlé dans le port *80/tcp*). A noter qu'il est également possible de croiser les journaux d'activités du *proxy* avec ceux de l'authentification YaCaP pour obtenir une correspondance directe entre usage et traçage nominatif.

L'utilisation de *firewall* en entrée/sortie de l'infrastructure captive apporte un niveau de sécurité supérieur (plus de possibilités de filtrage et d'inspection sur un *firewall* qu'avec de simples *ACLs*) mais également une plus grande liberté pour les administrateurs d'établissements qui peuvent gérer eux-mêmes cette sécurité (le portail captif n'a alors plus qu'un rôle « d'interrupteur » sur authentification).

La connexion d'un système de détection d'intrusions (*IDS*) en ce point stratégique permet la détection « au plus tôt » des machines clientes malveillantes et ainsi de pouvoir remonter à l'utilisateur un éventuel problème de sécurité sur son poste.

Les serveurs captifs sont tout particulièrement exposés car accessibles directement par les clients. Ils nécessitent d'être protégés pour garantir leur intégrité. L'utilisation des *netfilter/iptables* au travers de l'outil d'administration *Vuurmuur* (cf. Annexe) disponibles sur *Linux* permettent de spécifier les flux à autoriser ou non entre les clients et les serveurs. L'ouverture sur les serveurs captifs des ports *TCP* et *UDP* strictement nécessaires et suffisants garantit la sécurité des services non disponibles pour les clients.

6 Utilisation sur réseau régional Lothaire

Le service de portail captif YaCaP est en production pour les trois Universités nancéennes, le CROUS de Nancy-Metz et d'autres établissements indépendants, soit 7 entités au total, depuis le printemps 2006.

Pour répondre au besoin de disponibilité et supporter le nombre d'utilisateurs simultanés annoncé par les partenaires (1500 utilisateurs universitaires et 8000 utilisateurs CROUS), l'infrastructure actuelle compte 6 serveurs Linux (3 fonctions x 2 pour la redondance = 6 serveurs) et 3 routeurs captifs pour un total de 38 *vlans* métropolitains et 13 *vlans* distants. L'implémentation actuelle de YaCaP sur Lothaire est donnée sur la Figure 10.

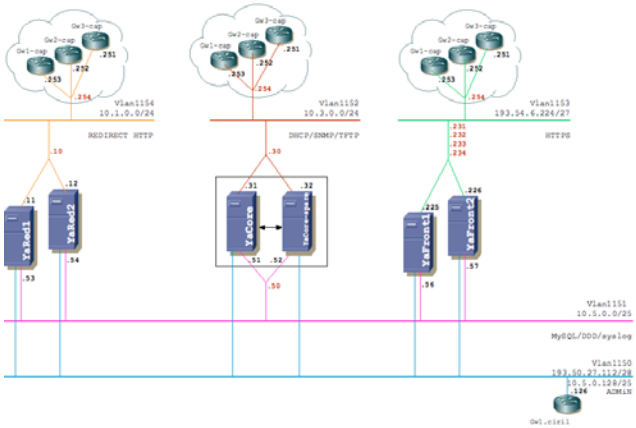

Figure 10: Infrastructure YaCaP pour Lothaire

L'architecture mise en oeuvre peut être synthétisée par :

- ✔ la séparation des trois fonctions : coeur, redirection et frontal ;
- ✔ la séparation des populations sur trois routeurs ;
- ✔ la séparation des fonctions sur des *subnets* différents ;
- ✔ la haute disponibilité de ces trois fonctions par l'utilisation du doublement et de la répartition de charge ;
- ✔ et la séparation des *subnets* côté client et côté administration des serveurs.

On observe aujourd'hui des pics de connexions à plus de 600 utilisateurs simultanés avec près de 10000 opérations de connexion / déconnexion sur 24 heures (ces chiffres augmentant de mois en mois). Certains *vlans* (CROUS) connaissent une activité quasi continue sur 24 heures.

Avec l'implémentation hautement disponible qui a été faite pour Lothaire et la supervision intégrée à YaCaP, la gestion quotidienne se réduit à son strict minimum : consultation des graphiques, vérification des traces, mise à jour sécurité des serveurs, déploiements de nouveaux réseaux captifs.

Dans le cadre du regroupement des Universités nancéiennes (Nancy-Université), YaCaP fait partie intégrante des applications mutualisées et assure la convergence des accès réseaux aux utilisateurs lorrains.

Annexe des outils libres utilisés

Perl

`http://www.perl.org/`

Apache

`http://httpd.apache.org/`

*mod_*perl

`http://perl.apache.org/`

MySQL

`http://www.mysql.com/`

OpenLDAP

`http://www.openldap.org/`

IPVS

`http://www.linuxvirtualserver.org/software/ipvs.html`

Keepalived

`http://www.keepalived.org/`

Netfilter/iptables

`http://www.netfilter.org/`

`http://www.kernel.org/`

Vuurmuur

`http://www.vuurmuur.org/`

`http://vuurmuur.sourceforge.net/`

DHCPd ISC

`http://www.isc.org/sw/dhcp/`

TFTPd

`http://www.kernel.org/pub/software/network/tftp/`

ProFTPD

`http://www.proftpd.org/`

Cfengine

`http://www.cfengine.org/`

Subversion

`http://subversion.tigris.org/`