Benjamin Gaillard
Lionel Imbs
Guillaume Lacroix

21 février 2007

Mobilité et Réseaux Sans Fil

CWP : portail web captif

Rapport

Master 2 RIA

ulp
UNIVERSITÉ LOUIS PASTEUR STRASBOURG

Table des matières

1

Introduction

Pour autoriser des clients à se connecter à un réseau sans fil, il existe plusieurs alternatives. L'une d'entre-elles est le portail web captif qui a l'avantage de donner accès à une partie du réseau (pouvant se limiter au portail lui-même uniquement) avant que l'authentification n'ait lieu. Cela autorise la mise en place d'un accès pour des personnes n'ayant pas besoin de connaître leurs informations d'authentification avant de pouvoir se connecter. Elles peuvent ainsi créer leur compte sur place, valider une charte d'utilisation, obtenir des informations pour configurer leur poste client, etc...

Au jour d'aujourd'hui il est nécessaire de prendre en compte le protocole IPv6 qui remplacera l'actuel IPv4 dans un avenir proche. Les solutions de portail web captif existantes ne proposent pas un tel support. Il est donc nécessaire de modifier l'existant ou de créer un nouveau portail. Nous avons choisi cette dernière solution car elle permet une prise en charge totalement intégrée d'IPv6. De plus, vous avons pu créer un portail gérant plus de services que la plupart des solutions existantes actuellement.

Nous allons donc présenter dans ce rapport le portail web captif que nous avons écrit. Nous l'avons appelé CWP, pour *Captive Web Portal*.

2

Fonctionnalités

Dans cette partie, nous allons présenter les fonctionnalités que nous avons implémentées dans CWP ainsi que les choix techniques que nous avons eus à effectuer.

2.1 Implémentation

CWP a été écrit en PHP 5. Notre choix s'est porté sur ce langage de script pour plusieurs raisons :

- possibilité d'écrire du code orienté objet ;
- omniprésence du langage : du fait de l'utilisation très répandue de PHP (une grande partie des sites web dynamiques), ce langage est facile d'accès et son installation doit être aisée quel que soit le système utilisé ;
- intégration à Apache : contrairement aux scripts CGI (Perl, Bash...), PHP peut être utilisé sous forme d'extension à Apache ce qui lui permet d'être résident en mémoire ; il n'est pas entièrement rechargé à chaque exécution d'un script. L'avantage direct est la possibilité d'utiliser CWP sur des systèmes limités en ressources tels que des routeurs.

De plus, CWP adopte une architecture modulaire : il peut ainsi être facilement étendu et/ou porté sur d'autres systèmes, en définissant juste de nouvelles classes basées sur les interfaces existantes. Il est notamment possible de rajouter des services, des méthodes d'authentification, le support d'un autre firewall, etc., de manière simple et rapide. Il est même prévu de pouvoir rajouter de nouveaux protocoles (ceux actuellement gérés étant Ethernet, IPv6 et IPv4). CWP est donc prêt pour de futures évolutions.

2.2 Sécurité

L'implémentation de CWP a été réalisée avec la sécurité comme facteur primordial. Le portail étant exécuté à travers Apache et devant avoir accès aux tables du *firewall*, il était nécessaire de trouver une solution qui permette la manipulation de ces tables sans avoir besoin de faire tourner le portail sous l'identité du superutilisateur *root*.

Pour ce faire, un *wrapper* a été écrit. Ce dernier permet d'exécuter les commandes iptables, *ip6tables* et *ebtables* sous l'identité d'Apache mais n'autorisent la manipulation que de certaines tables : celles dont le nom commence par « cwp_ ». Le portail peut ainsi modifier ses propres règles tout en n'ayant pas accès aux autres tables du pare-feu.

2.3 Services utilisés

CWP nécessite les éléments suivants pour fonctionner :

- un système d'exploitation basé sur le noyau Linux : le fonctionnement du portail repose sur Netfilter — commandes iptables et ip6tables — pour les autorisation d'accès et la redirection sur la page web du portail ; cependant CWP peut être aisément porté sur un autre système du fait de son implémentation modulaire ;
- ebtables, si l'on a l'intention d'utiliser un pont réseau (voir la section suivante pour plus de détails) ;
- iproute2 pour Linux (successeur d'ipconfig), pour la récupération et l'application de paramètres liés aux interfaces réseau ;
- Apache, avec les modules mod_rewrite et si possible mod_ssl ;
- PHP 5 : les fonctionnalités objet de la version 5 sont requises, les version antérieures ne sont pas supportées. Le module POSIX est obligatoire ainsi que le module PECL RADIUS si cette méthode d'authentification est utilisée ;
- optionnellement, les serveurs Bind (DNS), RADVD (annonce de routeur IPv6) et DHCPD (DHCP pour IPv4), suivant les besoins.

CWP est capable de configurer et de lancer automatiquement des serveurs additionnels. Ceux-ci incluent un serveur DNS (servant de cache et permettant de résoudre l'adresse spéciale portal.cwp), un annonceur de routeur IPv6 pour l'autoconfiguration des adresses IPv6 et un serveur DHCP pour l'attribution des adresses IPv4. Si les exécutables de ces serveurs sont présents sur le système, CWP les utilise par défaut.

2.4 Support réseau

Il est possible de faire fonctionner CWP sur un pont (*bridge*), permettant ainsi de laisser la tâche de l'attribution des adresses à un autre routeur déjà présent sur le réseau. À noter que cela peut être fait de manière indépendante pour IPv4 et IPv6 ; par exemple, il est tout à fait possible que l'adresse IPv4 soit attribuée par notre propre serveur DHCP mais que l'adresse IPv6 soit annoncée par un autre routeur, y compris si un autre serveur DHCP est déjà présent.

Le portail peut être accédé en utilisant son adresse IP ou l'adresse spéciale portal.cwp (si un serveur DNS est disponible et activé). Si l'authentification n'a pas encore eu lieu, le client sera de toute manière automatiquement redirigé vers le portail s'il tente d'accéder à un autre site.

Dans le cas où c'est CWP qui attribue les adresses, il va se baser sur les adresses déjà définies sur l'interface connectée au sous-réseau qu'il devra gérer. Si une adresse n'est pas encore définie, une adresse automatique sera automatiquement ajoutée (adresse privée pour IPv4, adresse *site-local* pour IPv6, générées aléatoirement pour éviter une éventuelle collision avec une adresse déjà existante). Si ce n'est pas ce que l'on souhaite, il suffit de définir manuellement les adresses souhaitées avant le démarrage de CWP.

Dans le cas où un pont est utilisé, celui-ci devra utiliser deux adresses pour chaque protocole : la première définie sera considérée comme étant celle du réseau interne, la seconde étant utilisée pour le sous-réseau dont le portail a la charge.

2.5 Filtrage et redirection

Par défaut, CWP n'autorise l'accès qu'aux services suivant avant authentification, le reste du trafic étant purement et simplement supprimé :

- sollicitation de voisin et de routeur IPv6 ;
- requêtes DHCP ;
- serveur(s) DNS ;
- le serveur web (HTTP et HTTPS) du portail.

L'accès au service web (HTTP) de n'importe quelle adresse autre que celle du portail génèrera une redirection vers le portail. Ainsi chaque tentative d'accès à un site Internet sera interceptée si l'authentification n'a pas encore été réalisée avec succès.

Il est possible de configurer CWP pour autoriser d'autres services sur d'autres machines avant que l'authentification n'ait eu lieu. Une fois celle-ci passée avec succès, l'accès au reste du réseau est autorisé. Pour configurer de manière fine le filtrage après authentification, il convient d'utiliser les commandes iptables et ip6tables, cela n'étant pas géré par CWP : en effet, cela serait redondant d'autant que l'utilisation des commandes sus-mentionnées permettent la réalisations de configurations bien plus puissantes et personnalisées, gérer toutes les fonctionnalités de Netfilter n'était pas le rôle de CWP.

2.6 Installation

CWP étant basé sur les GNU Autotools, l'installation relève d'un classique : `./configure && make && make install`.

Une fois le paquet installé, il convient de configurer Apache pour qu'il puisse fonctionner avec. Un fichier d'exemple de configuration est livré avec. De même, si l'on veut pouvoir lancer les services associés au démarrage, il est recommandé d'écrire les scripts correspondant. Cela étant spécifique à chaque distribution, le script n'a pas été créé.

Cependant, les fichiers nécessaires à un bon fonctionnement sous une distribution Gentoo Linux sont fournis, incluant l'*ebuild*. En utilisant ce fichier, on peut donc installer CWP en tapant la commande « `emerge net-misc/cwp` ».

Il est préférable d'ajuster les paramètres (*USE flags*) suivant ce que l'on désire installer/utiliser. L'*ebuild* se charge lui-même de configurer Apache et de créer une clé autosignée pour le fonctionnement d'Apache avec SSL (serveur web HTTPS).

Note : il est recommandé d'utiliser un accélérateur PHP tel qu'eAccelerator pour une exécution plus rapide des scripts du portail.

3

Fichiers

Dans cette partie, nous allons décrire les différents fichiers composant le projet : l'organisation de l'archive, les fichiers source PHP et les fichiers de configuration.

3.1 Organisation de l'archive

Les sources de CWP sont réparties dans différents répertoire suivant leurs fonctions. Il s'agit des répertoires suivant :

- doc-fr : la documentation française (ce rapport) ;
- etc : fichiers de configuration et scripts de démarrage génériques ;
- gentoo : l'*ebuild*, la configuration d'Apache et le script de démarrage pour une distribution Gentoo ;
- htdocs : les fichiers utilisés pour l'interface web (feuille de style, images) ;
- php : les sources du portail, en PHP 5 orienté objet ;
- wrapper : les *wrappers* pour iptables, ip6tables et ebtables (voir la section sur la sécurité pour connaître leur utilité).

3.2 Fichiers source PHP

Les sources PHP sont organisées de telle façon qu'à une classe « de base » corresponde un fichier. Par classe « de base », nous entendons les classes qui n'héritent pas d'une autre. Ainsi, par exemple, la classe `Portal` réside seule dans un fichier et les classes `Address`, `Ipv6Address`, `Ipv4Address` et `EthernetAddress` se situent dans le même fichier. Tous ces fichiers se trouvent dans le répertoire `php`.

Voici une liste des fichiers source avec les classes qu'ils contiennent ainsi qu'une

description succinte de chacune :

- address.php : contient les classes qui permettent d'effectuer des opérations sur les adresses IP et Ethernet : `Address`, `Ipv6Address`, `Ipv4Address` et `EthernetAddress` ;
- auth.php : déclare l'interface `iAuthenticater` et les classes `AuthTest` et `Radius` gérant les méthodes d'authentification ;
- base.php : héberge la classe `Base` permettant de trouver et d'inclure les autres fichiers PHP ;
- config.php : habrite la classe `Config` qui offre des méthodes permettant de connaître des valeurs de configuration. Cette classe lit le fichier de configuration principal /etc/cwp/config.php ;
- firewall.php : contient l'interface `iFirewall` et la classe `Iptables` chargée de manipuler les règles du pare-feu ;
- init.php : héberge la classe `InitScript` qui est chargée de démarrer et d'arrêter CWP et les services qu'il gère ;
- interface.php : déclare la classe `NetworkInterface` qui va permettre l'énumération et la configuration des interfaces réseau du système. Elle permet notamment de connaître les adresses IP utilisées ainsi que l'adresse MAC correspondant à chaque interface ;
- portal.php : habrite la classe `Portal` chargée de l'interface utilisateur. Cette classe génère une page web conforme aux standards XHTML 1.1 et CSS 2 établis par le consortium W3C ;
- protocol.php : contient la classe `Protocol` énumérant les protocoles reconnus par le portail. À ce jour, il s'agit d'IPv6, IPv4 et Ethernet ;
- service.php : héberge les classes `Service` (classe de base fournissant des méthodes communes aux classes dérivées), `ClientsFile` (création si nécessaire du fichier avec les bons droits listant les clients authentifiés et leurs adresses respectives), `Radvd` (annonceur de routeur IPv6), `Dhcpv4d` (serveur DHCP IPv4) et `Dns` (serveur DNS).

Deux fichiers ne contiennent pas de classe. Ils n'ont pour utilité que d'instancier une classe précise. Ils servent d'accesseurs au code PHP depuis l'extérieur. Ces fichiers sont :

- php/cwp.php : ce fichier est celui qui sera exécuté par Apache. Il instancie la classe `Portal` qui sera chargée d'afficher l'interface utilisateur ;
- etc/init.php : utilisé pour le démarrage des services, ce fichier instancie la classe `InitScript`.

Pour finir, il existe encore le fichier dirs.php créé lors de la compilation de l'archive. Il ne contient pas de classe mais déclare quelques variables indiquant les chemins d'installation du paquet.

3.3 Fichier de configuration

Le fichier de configuration, installé à l'emplacement /etc/cwp/config.php, permet de régler toutes sortes de paramètres de CWP. Voici le contenu initial de ce fichier, listant les différentes options disponibles avec pour chacun une brève description :

```
<?php

# Authentivation service to use.  Default is 'AuthTest' which is a fake one
# accepting only the "luke"/"ifeeltheforce" (without quotes) login/password
```

```
# pair.  It is of course strongly advised to use a better service.
# Possible values: "AuthTest", "Radius"
#$AUTH = 'AuthTest';

# Maximum delay a client remains authenticated without accessing the portal.
# This is expressed in seconds.  Default: 60.
#$AUTH_DELAY = 60;

# Parameters for the RADIUS authentication method.
#$RADIUS_SERVER = 'localhost';
#$RADIUS_PORT   = 0;
#$RADIUS_SECRET = '';

# The interface connected to the clients (should be your wireless card).
# If not specified, try br0, wlan0, ath0, eth1, eth0 (in that order).
#$INT_INTERFACE = 'wlan0';

# The interface we intercept trafic from.
# Should be a bridge interface or the same as $INT_INTERFACE above.
#$INTERFACE = $INT_INTERFACE;

# Wether to use NAT for connections coming from the managed network.
# This is enabled by default.
#$USE_NAT = TRUE;

# Wether to filter accesses by MAC address (enabled by default).
# Note: this requires the iptables MAC match module.
#$USE_MAC = TRUE;

# Wether to auto assign IP addresses to the interface.
# Enabled by default if not operating on a bridge.
#$AUTO_IPV6 = TRUE;
#$AUTO_IPV4 = TRUE;

# Wether to use a bridge for address assignment.
# Disabled by default unless operating on a bridge.
#$BRIDGE_IPV6 = FALSE;
#$BRIDGE_IPV4 = FALSE;

# Services: enable or disable a service explicitely.
# They are all enabled by default if available and needed.
#$RADVD   = TRUE; # IPv6 router advertisement
#$DHCPV4D = TRUE; # IPv4 DHCP
#$DNS     = TRUE; # DNS proxy (necessary for valid HTTPS)

# Firewall engine to use.  Only "Iptables" is possile until CWP gets ported
# to other architectures.  DON'T SET THIS VALUE!
#$FIREWALL = 'Iptables';

?>
```

4

Conclusion

Ce projet nous a permis d'appréhender les diverses problématiques liées à la mise en place d'un portail web captif : l'implémentation d'un ensemble de classes en PHP, la gestion de services couramment utilisés dans ce genre d'architecture (serveur DHCP, annonceur de routeur IPv6, etc.), la manipulation des règles d'un firewall aussi bien en IPv6 qu'en IPv4, l'utilisation d'un pont pour certains services...

CWP a de plus été écrit avec l'intention que son développement soit poursuivi à l'avenir. Son architecture est résolument ouverte vers l'implémentation de nouvelles classes permettant de gérer davantage de services, méthodes d'authentification et protocoles.