Adaptations de FreeRadius pour Debian Jessie

Jusqu'en mars 2015 :

Switch → FreeRadius filaire (sur le serveur radius.adm.crans.org) → Auth.py

: Applique le site "filaire",

fait appel aux fonction authorize et post authentication de auth.py

Borne Wifi → FreeRadius wifi (sur le serveur eap.wifi.crans.org) → Auth.py

: Applique le site "wifi",

fait appel aux fonction authorize et post authentication de auth.py

Ancien mode de fonctionnement

```
root@pea:/etc/freeradius# ls
acct_users                  au                  dictionary              ldap.attrmap    radiusd.conf            sqlippool.conf
attrs                       certs               dynamic_clients.conf    modules         radiusd.conf.old        templates.conf
attrs.access_challenge      certs_28.12.13      eap.conf                policy.conf     rlm_python_wifi.conf.old users
attrs.access_reject         certs_backup        experimental.conf       policy.txt      sites-available         wifi.key.old
attrs.accounting_response   certs_gordon.old    hints                   preproxy_users  sites-enabled
attrs.pre-proxy             clients.conf        huntgroups              proxy.conf      sql.conf
root@pea:/etc/freeradius#
```

```
# Configuration for the Python module.
#
#

python crans_fil {
>---mod_instantiate = freeradius.auth
>---func_instantiate = instantiate

    # Spécifique au filaire: accepte direct
>---mod_authorize = freeradius.auth
>---func_authorize = authorize_fil
....
    # Renseigne le vlan
    # remplacer par dummy_fun pour ignorer le tagging de vlan
>---mod_post_auth = freeradius.auth
>---func_post_auth = post_auth_fil

    # Que faire avant de quitter
>---mod_detach = freeradius.auth
>---func_detach = detach

    # Le reste est dumb et inutile
>---mod_accounting = freeradius.auth
>---func_accounting = dummy_fun

>---mod_pre_proxy = freeradius.auth
>---func_pre_proxy = dummy_fun

>---mod_post_proxy = freeradius.auth
>---func_post_proxy = dummy_fun

>---mod_recv_coa = freeradius.auth
>---func_recv_coa = dummy_fun

>---mod_send_coa = freeradius.auth
>---func_send_coa = dummy_fun
}
```

```
##########################################################################
#
#  Authentification filaire du crans
#
##########################################################################

server filaire {
  authorize{
    preprocess
    crans_fil
  }

  authenticate{
      crans_fil
  }

  post-auth{
      crans_fil
  }
}
~
```

Depuis mars 2015 :

Switch

Borne Wifi

FreeRadius unifié (sur eap et radius)

Le serveur radius applique un sites filaire ou wifi suivant le client qui parle , on crée pout cela un nouveau sites qui détermine quel sous-site applique (filaire/wifi)

(système paresseux, si c'est pas un switch c'est une borne)

Auth.py

Les sites available sont dans /usr/scripts

```
.-(14:40:11)-(/usr/scripts/freeradius/sites-available)----------------
[`---> ls
dynamic_clients  filaire  inner-tunnel  wifi
```

```
# Configuration for the Python module.
#
#

python crans_nas {
>---mod_instantiate = freeradius.auth
>---func_instantiate = instantiate

    # Spécifique NAS : rempli le mdp
>---mod_authorize = freeradius.auth
>---func_authorize = authorize_nas

    # Que faire avant de quitter
>---mod_detach = freeradius.auth
>---func_detach = detach

    # Le reste est dumb et inutile
>---mod_post_auth = freeradius.auth
>---func_post_auth = dummy_fun

>---mod_accounting = freeradius.auth
>---func_accounting = dummy_fun

>---mod_pre_proxy = freeradius.auth
>---func_pre_proxy = dummy_fun

>---mod_post_proxy = freeradius.auth
>---func_post_proxy = dummy_fun

>---mod_recv_coa = freeradius.auth
>---func_recv_coa = dummy_fun

>---mod_send_coa = freeradius.auth
>---func_send_coa = dummy_fun
}
```

```
# Configuration for the Python module.
#
#

python crans_fil {
>---mod_instantiate = freeradius.auth
>---func_instantiate = instantiate

    # Spécifique au filaire: accepte direct
>---mod_authorize = freeradius.auth
>---func_authorize = authorize_fil
••••
    # Renseigne le vlan
    # remplacer par dummy_fun pour ignorer le t
>---mod_post_auth = freeradius.auth
>---func_post_auth = post_auth_fil

    # Que faire avant de quitter
>---mod_detach = freeradius.auth
>---func_detach = detach

    # Le reste est dumb et inutile
>---mod_accounting = freeradius.auth
>---func_accounting = dummy_fun

>---mod_pre_proxy = freeradius.auth
>---func_pre_proxy = dummy_fun

>---mod_post_proxy = freeradius.auth
>---func_post_proxy = dummy_fun

>---mod_recv_coa = freeradius.auth
>---func_recv_coa = dummy_fun

>---mod_send_coa = freeradius.auth
>---func_send_coa = dummy_fun
}
```

```
# Configuration for the Python module.
#
#

python crans_wifi {
>---mod_instantiate = freeradius.auth
>---func_instantiate = instantiate

    # Spécifique au WiFi : rempli le mdp
>---mod_authorize = freeradius.auth
>---func_authorize = authorize_wifi
••••
    # Renseigne le vlan
    # remplacer par dummy_fun pour ignorer le tagging de
>---mod_post_auth = freeradius.auth
>---func_post_auth = post_auth_wifi

    # Que faire avant de quitter
>---mod_detach = freeradius.auth
>---func_detach = detach

    # Le reste est dumb et inutile
>---mod_accounting = freeradius.auth
>---func_accounting = dummy_fun

>---mod_pre_proxy = freeradius.auth
>---func_pre_proxy = dummy_fun

>---mod_post_proxy = freeradius.auth
>---func_post_proxy = dummy_fun

>---mod_recv_coa = freeradius.auth
>---func_recv_coa = dummy_fun

>---mod_send_coa = freeradius.auth
>---func_send_coa = dummy_fun
}
```

```python
@radius_event
def authorize_wifi(data):
    """Section authorize pour le wifi
    (NB: le filaire est en accept pour tout le monde)
    Éxécuté avant l'authentification proprement dite. On peut ainsi remplir les
    champs login et mot de passe qui serviront ensuite à l'authentification
    (MschapV2/PEAP ou MschapV2/TTLS)"""

    items = get_machines(data)

    if not items:
        logger.error('No machine found in lc_ldap')
        return radiusd.RLM_MODULE_NOTFOUND

    if len(items) > 1:
        logger.warn('lc_ldap: Too many results (taking first)')

    machine = items[0]

    proprio = machine.proprio()
    if isinstance(proprio, lc_ldap.objets.AssociationCrans):
        logger.error('Crans machine trying to authenticate !')
        return radiusd.RLM_MODULE_INVALID

    for bl in machine.blacklist_actif():
        if bl.value['type'] in BL_REJECT:
            return radiusd.RLM_MODULE_REJECT
        # Kludge : vlan isolement pas possible, donc reject quand-même
        if not WIFI_DYN_VLAN and bl.value['type'] in BL_ISOLEMENT:
            return radiusd.RLM_MODULE_REJECT


    if not machine.get('ipsec', False):
        logger.error('WiFi auth but machine has no password')
        return radiusd.RLM_MODULE_REJECT

    password = machine['ipsec'][0].value.encode('ascii', 'ignore')

    # TODO: feed cert here
    return (radiusd.RLM_MODULE_UPDATED,
>-------     (),
        (
        ("Cleartext-Password", password),
        ),
      )
```

A partir de maintenant :

Fusion totale, c'est auth.py qui répond suivant la tête de la requète

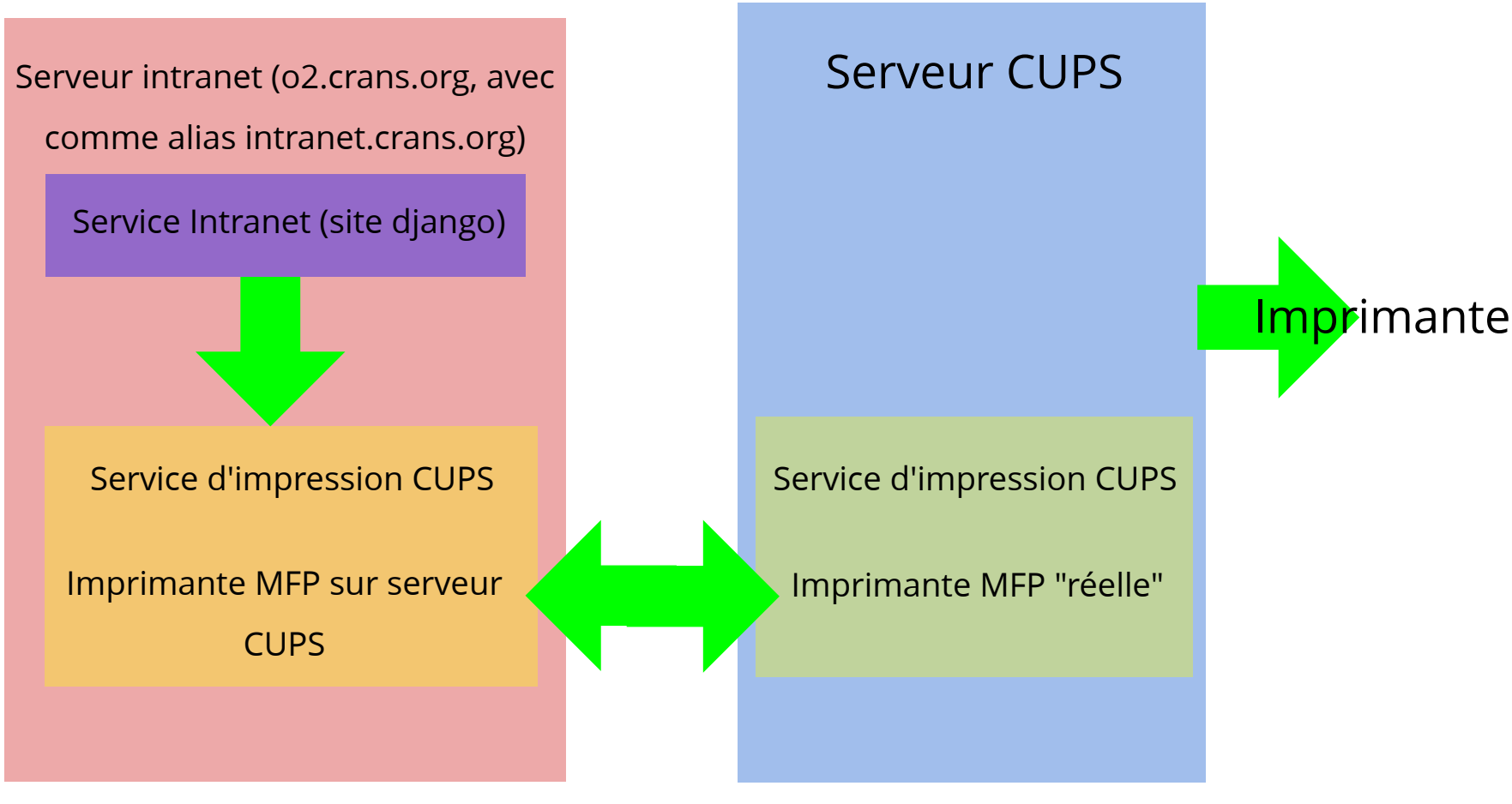
```
#######################################
#
#  Authentification filaire du crans
#
#######################################

server filaire {
  authorize{
    preprocess
    crans_unifie
  }

  authenticate{
      crans_unifie
  }

  post-auth{
      crans_unifie
  }
}
~
```

```
# Configuration for the Python module.
#
#

python crans_unifie {
        mod_instantiate = freeradius.auth
        func_instantiate = instantiate

    # Pour le authorize, c'est auth.py qui fait le tri maintenant
        mod_authorize = freeradius.auth
        func_authorize = authorize

    # Renseigne le vlan si necessaire
    # remplacer par dummy_fun pour ignorer le tagging de vlan
        mod_post_auth = freeradius.auth
        func_post_auth = post_auth

    # Que faire avant de quitter
        mod_detach = freeradius.auth
        func_detach = detach

    # Le reste sert à rien
        mod_accounting = freeradius.auth
        func_accounting = dummy_fun

        mod_pre_proxy = freeradius.auth
        func_pre_proxy = dummy_fun

        mod_post_proxy = freeradius.auth
        func_post_proxy = dummy_fun

        mod_recv_coa = freeradius.auth
        func_recv_coa = dummy_fun

        mod_send_coa = freeradius.auth
        func_send_coa = dummy_fun
}
```


debian

Système d'impression actuel et évolutions possibles

Configuration actuelle (depuis mars 2015)



Configuration envisagée