FreeRADIUS configuration

NTLM configuration

NTLM/MSCHAP configuration

```
mschap {
    with_ntdomain_hack = yes
        use_mppe = yes
        require_encryption = no
        require_strong = no
        pool {
                start = ${thread[pool].start_servers}
                min = ${thread[pool].min_spare_servers}
                max = ${thread[pool].max_servers}
                spare = ${thread[pool].max_spare_servers}
                uses = 0
                retry_delay = 30
                lifetime = 86400
                cleanup_interval = 300
                idle_timeout = 600
        }

        passchange {
        }

        # ntlm_auth = "/usr/bin/ntlm_auth --request-nt-key --username=%{%{Stripped-User-Name}:-%{%{User-Name}:-
None}} --challenge=%{%{mschap:Challenge}:-00} --nt-response=%{%{mschap:NT-Response}:-00}"

        winbind_username = "%{mschap:User-Name}"
        winbind_domain = "%{mschap:NT-Domain}"
}
```

LDAP configuration

LDAP configuration

```
ldap {
        server = 'ldap://AD_DOMAIN'
        port = 389
        identity = AD_USERNAME@AD_DOMAIN
        password = AD_PASSWORD
        base_dn = AD_BASE_DN
        sasl {
        }

        update {
                control:Password-With-Header        += 'userPassword'
                control:NT-Password                := 'ntPassword'
                reply:Reply-Message                := 'radiusReplyMessage'
                reply:Tunnel-Type                 := 'radiusTunnelType'
                reply:Tunnel-Medium-Type         := 'radiusTunnelMediumType'
                reply:Tunnel-Private-Group-ID        := 'radiusTunnelPrivategroupId'

                #  Where only a list is specified as the RADIUS attribute,
                #  the value of the LDAP attribute is parsed as a valuepair
                #  in the same format as the 'valuepair_attribute' (above).
                control:                        += 'radiusControlAttribute'
                request:                        += 'radiusRequestAttribute'
                reply:                                += 'radiusReplyAttribute'
        }

        edir = no
```

```
user {
        base_dn = "${..base_dn}"
        filter = "(sAMAccountName=%{%{Stripped-User-Name}:-%{User-Name}})"
        sasl {
        }
}

group {
        base_dn = "${..base_dn}"
        name_attribute = "CN"
        filter = '(objectClass=posixGroup)'
        membership_attribute = 'memberOf'
}

profile {
}

client {
        base_dn = "${..base_dn}"
        filter = '(objectClass=radiusClient)'
        template {
        }

        attribute {
                ipaddr                                  = 'radiusClientIdentifier'
                secret                                  = 'radiusClientSecret'
        }
}

accounting {
        reference = "%{tolower:type.%{Acct-Status-Type}}"

        type {
                start {
                        update {
                                description := "Online at %S"
                        }
                }

                interim-update {
                        update {
                                description := "Last seen at %S"
                        }
                }

                stop {
                        update {
                                description := "Offline at %S"
                        }
                }
        }
}

post-auth {
        update {
                description := "Authenticated at %S"
        }
}

options {
        chase_referrals = yes
        rebind = yes
        res_timeout = 10
        srv_timelimit = 3
        net_timeout = 1
        idle = 60
        probes = 3
        interval = 3
        ldap_debug = 0x0028
}
```

```
        tls {
        }

        pool {
                start = ${thread[pool].start_servers}
                min = ${thread[pool].min_spare_servers}
                max = ${thread[pool].max_servers}
                spare = ${thread[pool].max_spare_servers}
                uses = 0
                retry_delay = 30
                lifetime = 0
                idle_timeout = 60
        }
}
```

Unlang logic, sites-enabled/default

default

```
authorize {
        filter_username
        preprocess
        chap
        mschap
        digest
        suffix
        eap {
                ok = return
#               updated = return
        }

        files
        -sql
        expiration
        logintime
        pap
}

authenticate {
        Auth-Type PAP {
                pap
        }

        Auth-Type CHAP {
                chap
        }

        Auth-Type MS-CHAP {
                mschap
        }

        mschap
        digest
        ldap
        eap
}

post-auth {
    if (LDAP-Group == "Test") {
            update reply {
                       Tunnel-Type := "VLAN"
            Tunnel-Medium-Type := "IEEE-802"
                Tunnel-Private-Group-Id := "Test"
        }
    }
    elsif (LDAP-Group == "Test2") {
            update reply {
                Tunnel-Type := "VLAN"
            Tunnel-Medium-Type := "IEEE-802"
            Tunnel-Private-Group-Id := "Test2"
        }
    }
         else {
        update reply {
                        Tunnel-Type := "VLAN"
            Tunnel-Medium-Type := "IEEE-802"
                        Tunnel-Private-Group-Id := "Unknown"
        }
    }
}
```