Prefix and Source IP Authentication for Incoming VoIP Traffic

This document explains how to authenticate incoming VoIP traffic to a Cisco IP-to_IP gateway using RADIUS protocol and TCL scripting. In this example following components are used:

- **IP-to-IP Gateway.** This is a Cisco gateway device with IP-to-IP gateway software installed. It can be a Cisco 3660 or Cisco 3845 model *(IOS version >= 12.4)*.
- **TekRADIUS** *(Version >= 1.9)***.**

You can see network components in the diagram below:

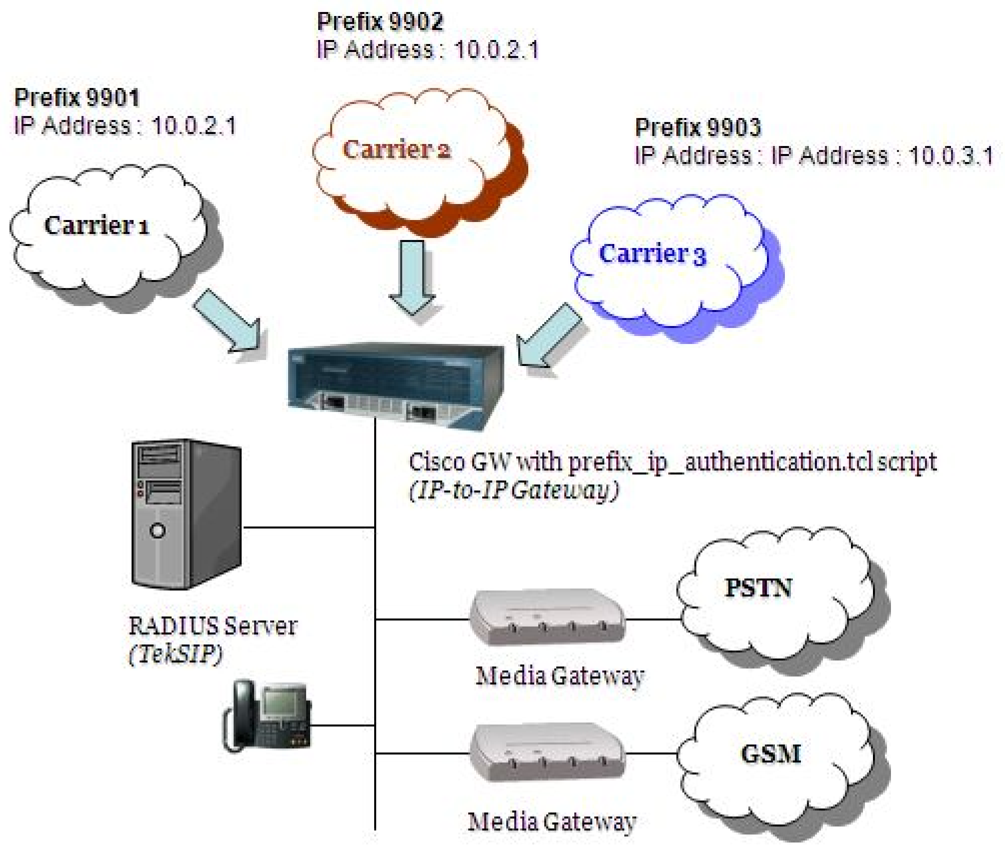
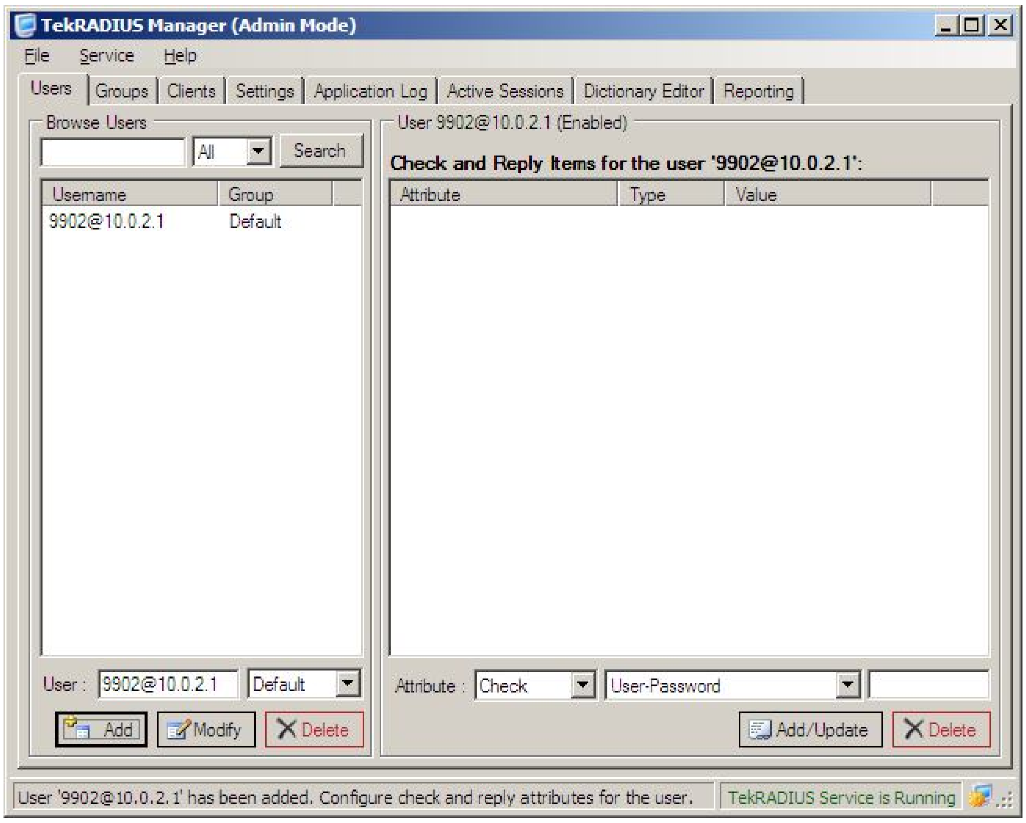


Figure - 1. Sample Topology

You can process VoIP calls *(H.323 or SIP)* using Cisco TCL scripting. You can authenticate and authorize incoming VoIP calls using RADIUS. TCL script will get IP address of the remote carrier and technical prefix from called number and concatenate them with a user defined character. Resulting string will be put in a RADIUS access-request packet as User-Name attribute and will be sent to TekRADIUS.

TekRADIUS Configuration

You'll need to add user profiles for remote carriers. User profiles must have a User-Name in Prefix + Concatenation Character + Carrier IP address format.



User 9902@10.0.2.1 (Enabled)

Check and Reply Items for the user '9902@10.0.2.1':

Username	Group
9902@10.0.2.1	Default

User : 9902@10.0.2.1 Default

Attribute : Check User-Password

User '9902@10.0.2.1' has been added. Configure check and reply attributes for the user.

TekRADIUS Service is Running

Figure - 2. Sample User Profile Entry

Do not forget to define you Cisco gateway as a RADIUS client in Clients tab. Re-start TekRADIUS after all settings done and saved.

Cisco Gateway Configuration

You need to have IP-to-IP Gateway software installed on Cisco Gateway. You need enter following configuration to communicate with RADIUS server:

```
aaa new-model
!
aaa authentication login default local
aaa authentication login h323 group radius
aaa authorization exec h323 group radius
! Add following line if you like to also get accounting data for VoIP sessions
aaa accounting connection h323 start-stop group radius
!
radius-server host 192.168.190.1 auth-port 1812 acct-port 1813 key secret
```

Configure Incoming dial peers for the carriers. SIP version 2 is used as VoIP protocol in our example:

```
dial-peer voice 50 voip
 description –- Carrier 1 - in –
 service prefix_ip_authentication
 incoming called-number 9901T
 session protocol sipv2
 codec transparent
!
```

```
dial-peer voice 60 voip
 description –- Carrier 2 - int –
 service prefix_ip_authentication
 incoming called-number 9902T
 session protocol sipv2
 codec transparent
!
dial-peer voice 70 voip
 description –- Carrier 3 - in –
 service prefix_ip_authentication
 incoming called-number 9903T
 session protocol sipv2
 codec transparent
```

Following TCL script is used for authorizing incoming VoIP calls *(call_route_app.1.0.tcl)*:

```
# Script Changed by: Yasin KAPLAN
# Script Version: 1.0
# Script Name: prefix_ip_authentication
# Script Lock Date: Mon Feb 15 14:06:00 2010
#------------------------------------------------------------------
# February 15th 2010, Yasin KAPLAN
#
# Copyright (c) 2010 by Yasin KAPLAN
# All rights reserved.
#------------------------------------------------------------------
#
# Description:
#
# This script allows the call from carrier authenticated by carrier prefix
# and IP address of the remote system.
#
#------------------------------------------------------------------
#

proc init_perCallVars { } {
    global disconnect_cause

    set disconnect_cause 0
}

proc act_Setup { } {
    global carrierid
    global carrierip
    global destination
    global account

    init_perCallVars

    set destination [infotag get leg_dnis]

    set carrierid [string range $destination 0 3]
    set carrierip [infotag get leg_remoteipaddress leg_incoming]
    set carrierid "$carrierid@$carrierip"

    set account [infotag get leg_username leg_incoming]

    aaa authenticate $carrierid ""
}

proc act_Authenticated { } {
    global account
    global destination
    global disconnect_cause

    set status [infotag get evt_status]
    puts "\n aaa authenticate Status=$status"
```

```
     if { $status == "au_000" } {

             set callInfo(accountNum) $account
             leg setup $destination callInfo leg_incoming

     } else {
           set disconnect_cause di_021
           act_SendCauseCode
          fsm setstate CALLDISCONNECT
     }
}

proc act_CallSetupDone { } {
    global creditTime
    global disconnect_cause

    set status [infotag get evt_status]

    puts "\t\t******* act_CallSetupDone: $status"

    if {$status == "ls_000"} {

        return
    }

    # leg setupFail
      set disconnect_cause [infotag get evt_last_disconnect_cause]
      act_SendCauseCode
      fsm setstate CALLDISCONNECT
}

proc act_SendCauseCode { } {
    global disconnect_cause

    puts "\t\t***** DISCONNECT CAUSE CODE: $disconnect_cause"

    set cause [split $disconnect_cause "_"]

    leg disconnect leg_incoming [lindex $cause 1]

}

proc act_ConnectionDestroy { } {
    global disconnect_cause

    set disconnect_cause [infotag get evt_last_disconnect_cause]
    connection destroy con_all
}

proc act_Cleanup { } {
    call close
}

requiredversion 2.0

#----------------------------------
#   State Machine
#----------------------------------
  set fsm(any_state,ev_disconnected)          "act_Cleanup           same_state"
  set fsm(any_state,ev_disconnect_done)       "act_Cleanup           same_state"

  set fsm(CALL_INIT,ev_setup_indication)      "act_Setup             AUTHENTICATE"

  set fsm(AUTHENTICATE,ev_authenticate_done)  "act_Authenticated     PLACECALL"

  set fsm(PLACECALL,ev_setup_done)            "act_CallSetupDone     CALLACTIVE"

  set fsm(CALLACTIVE,ev_disconnected)         "act_SendCauseCode     CALLDISCONNECT"

  set fsm(CALLACTIVE,ev_disconnected)         "act_ConnectionDestroy CONNDESTROY"
  set fsm(CONNDESTROY,ev_destroy_done)         "act_SendCauseCode      CALLDISCONNECT"

  set fsm(CALLDISCONNECT,ev_disconnected)      "act_Cleanup            same_state"
  set fsm(CALLDISCONNECT,ev_destroy_done)      "act_Cleanup            same_state"
  set fsm(CALLDISCONNECT,ev_disconnect_done)   "act_Cleanup            same_state"

  fsm define fsm CALL_INIT
```

Enter following configuration to define this script as an application:

```
application
 service carrier_routing flash:call_route_app.1.0.tcl
```

References

1. Cisco IP-to-IP Gateway Configuration
   http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/callc_c/h323_c/ipipgw/ipgw.htm
2. Cisco TCL Scripting
   http://www.cisco.com/en/US/products/sw/iosswrel/ps5207/products_feature_guide09186a00801a75a7.html
3. TekRADIUS
   http://www.tekradius.com/
4. Microsoft SQL Server
   http://www.microsoft.com/sql/default.mspx