<http://www.routingloops.co.uk/cisco/tacacs-on-ubuntu-14-04-lts/>

# TACACS+ ON UBUNTU 14.04 LTS

[25/10/2014](http://www.routingloops.co.uk/cisco/tacacs-on-ubuntu-14-04-lts/) [JONATHAN](http://www.routingloops.co.uk/author/admin/) [11 COMMENTS](http://www.routingloops.co.uk/cisco/tacacs-on-ubuntu-14-04-lts/#comments)

[](http://www.routingloops.co.uk/wp-content/uploads/2014/10/ubuntu-logo32.png)In this post I will be describing the steps required to install TACACS+ on Ubuntu 14.04 LTS. I will be compiling the TACACS+ daemon with ACL support for additional security.

I will also run through the basic configuration of Cisco devices to use TACACS+ for Authentication, Authorisation and Accounting.

##### History of TACACS+

Terminal Access Controller Access-Control System (TACACS) is a protocol providing a centralised server for remote Authentication, Authorisation and Accounting of network devices. The TACACS protocol was originally developed in 1984 by BBN Technologies for administering MILNET, which ran unclassified network traffic for DARPA and was first formally defined in RFC 927.

Cisco Systems added TACACS support to its network devices in the late 1980’s and went on to add a number of extensions to the protocol, most notably Extended TACACS (XTACACS) and TACACS plus (TACACS+).

Although derived from TACACS the TACACS+ protocol is an entirely new protocol and is not backwards compatabile with TACACS and XTACACS. TACACS+ uses TCP as its transport layer protocol, typically using port number 49. Unlike RADIUS TACACS+ separates the Authentication and Authorisation functions making it more flexible for network administration.

TACACS+ allows you to set granular access policies for users and groups, commands, location, subnet, or even device type. The TACACS+ protocol also provides detailed logging of users and what commands have been run on specific devices.

##### Installation

We will be using TACACS+ daemon provided by Shrubbery Networks for our installation of TACACS+. All commands will be run as root and therefore the first step is to use sudo to log into your root account:



|  |  |
| --- | --- |
| 1 | sudo su |

###### Install Dependencies

As always before we can start the installation we need to update our repositories and install the package dependencies.

In this case the dependencies are as follows:

* gcc
* bison
* flex
* libwrap0-dev

Before installing the dependencies you can check if they are already installed using the dpkg -s command:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | root@routingloop:~# dpkg -s gcc bison flex  dpkg-query: package 'gcc' is not installed and no information is available    dpkg-query: package 'bison' is not installed and no information is available    dpkg-query: package 'flex' is not installed and no information is available  Use dpkg --info (= dpkg-deb --info) to examine archive files,  and dpkg --contents (= dpkg-deb --contents) to list their contents.  root@routingloop:~# |

If the dependeincies are not installed issue the following command:



|  |  |
| --- | --- |
| 1 | apt-get update && apt-get install -y gcc make flex bison libwrap0-dev |

###### Install TACACS+

We will be using the TACACS+ daemon from Shrubbery Networks. At the time of writing the latest stable version is tacacs+-F4.0.4.26.

Download and extract the source code for TACACS+:



|  |  |
| --- | --- |
| 1 | wget ftp://ftp.shrubbery.net/pub/tac\_plus/tacacs+-F4.0.4.26.tar.gz && tar zxvf tacacs+-F4.0.4.26.tar.gz |

Change directory to the newly created tacacs+-F4.0.4.26 directory and view the installation file to see the installation options:



|  |  |
| --- | --- |
| 1  2 | cd tacacs+-F4.0.4.26  less INSTALL |

Sample output of the installation instructions:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | This is a modified version of Cisco's tacacs+ (tac\_plus) "developer's  kit."    Quick Installation Guide (an example):    1) ./configure [--prefix=]     By default, All tac\_plus crud will be installed under /usr/local.     This can be overridden with the --prefix option.  E.g.:    ./configure --prefix=/usr/pkg       see ./configure --help for other configure options.    2) make install    3) it may be necessary, or you may want to, add tacacs to your /etc/services     file in order for tacacs to work properly.  eg:       tacacs tcp/49    4) read the tac\_plug(8) manual page    5) Send any bugs, suggestions or updates to tac\_plus@shrubbery.net.     See the web page at http://www.shrubbery.net/tac\_plus.    Prerequisites for building:  - Wietse Venema's TCP wrappers library |

If you would like to check all the options available when compiling the TACACS+ daemon run ./configure –help.

In our case the most of the defaults are suitable, the only changes I will make are to change the installation directory (using /usr rather than /usr/local), enable ACL support and enabling per user enable passwords using the following command:



|  |  |
| --- | --- |
| 1 | ./configure --prefix=/usr --enable-acls --enable-uenable && make install |

After installing TACACS+ the binary files will be available in /usr/bin/ (if you chose to install into the default location the files will be in /usr/local/bin):



|  |  |
| --- | --- |
| 1  2  3 | root@tacacs:~/tacacs+-F4.0.4.26# ls /usr/bin/tac\*  /usr/bin/tac  /usr/bin/tac\_plus  /usr/bin/tac\_pwd  root@tacacs:~/tacacs+-F4.0.4.26 |

* tac\_plus is the TACACS+ daemon
* tac\_pwd is used to generate a Data Encryption Standard (DES) or Message-Digest 5 (MD5) hash from a clear text password. Note, if you want to use MD5 you need to use the -m option when generating the password.

###### Fix Library Links

Finally we need to ensure that the correct links to the libraries required are installed to ensure that the TACACS+ daemon starts correctly.

Tod do this add /usr/lib (if you did not change the path of the install using –prefix=/usr then add /usr/local/lib) to /etc/ld.so.conf:



|  |  |
| --- | --- |
| 1 | vi /etc/ld.so.conf |



|  |  |
| --- | --- |
| 1  2  3 | root@tacacs:/etc/tacacs# cat /etc/ld.so.conf  include /etc/ld.so.conf.d/\*.conf  /usr/lib |

Next reload the libraries using ldconfig:



|  |  |
| --- | --- |
| 1 | ldconfig |

##### Configure TACACS+

For this example I will be creating two groups called network\_admin and sys\_admin. The network\_admin group will have full privilege 15 rights on the router while the sys\_admin group will only have access to show commands, and be able to configure interfaces with the basic settings such as access vlan, trunk and description.

Each group will have an ACL applied, where the network\_admin group will have access to any device while the sys\_admin group will only be allowed to access specific devices.

Each user will have their own password and enable password. I will also show two methods for authenticating users:

1. Using a configured password and the tac\_pwd comand to encrypt it.
2. Using a user that is configured on the system and authenticating from the /etc/passwd file.

##### tac\_plus.conf file explanation

First we need to create the directory and tac\_plus.conf file where the TACACS+ configuration will be defined:



|  |  |
| --- | --- |
| 1  2  3  4 | mkdir /etc/tacacs  cd /etc/tacacs  touch tac\_plus.conf  chmod 755 tac\_plus.conf |

We will also need to create a file where the TACACS+ accounting logs will be sent to.



|  |  |
| --- | --- |
| 1  2 | mkdir /var/log/tac\_plus  touch /var/log/tac\_plus/tac\_plus.acct |

I will run through the different sections of the tac\_plus config file with a description of their purpose then I will provide a template that can be used to setup a basic server.

###### Encryption Key

The first thing that needs to be configured is encryption key used to encrypt packets between the daemon and clients. This key must match the key configured on the clients:



|  |  |
| --- | --- |
| 1  2  3 | # Encryption key    key = "tac\_test" |

###### Accounting

All accounting records are either written to a file, syslog at priority info, or both.In this example I will set the accounting records to be sent to both the file we created in /var/log/tac\_plus, as well as syslog (to disable sending to syslog and only send to the file simply uncomment the line specifying syslog with #):



|  |  |
| --- | --- |
| 1 | # Set where to send accounting records |



|  |  |
| --- | --- |
| 1  2 | accounting syslog;  accounting file = /var/log/tac\_plus/tac\_plus.acct |

###### Access Control Lists

Next we will create the ACL’s that we will use for the different groups. Access Control Lists can be defined to limit user’s, or group’s, login and/or enable access by daemon client IP address or hostname. An ACL is referenced by its name, but must be defined before it can be referenced. The ACL is a series of permit or deny statements applied to the source IP address that the daemon client used to connected to the daemon. If no entry of the acl matches a given address, the result is an implicit deny.

The default syntax for ACL’s is as follows:



|  |  |
| --- | --- |
| 1  2  3  4 | acl = name {  permission = regex  # implicit deny (ie: anything else)  } |

In our example I will allow the network\_admin group access from any source address, while the sys\_admin group will only be allowed access from a source of 10.10.10.250.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | # ACL for network\_admin group    acl = network\_admin {  # allow access to all devices  permit = .\*  # implicit deny (ie: anything else)          }    # ACL for sys\_admin group    acl = sys\_admin {  # allow access to 10.10.10.2 only  permit = ^10\.10\.10\.2$  # implicit deny (ie: anything else)          } |

###### Groups

Next we will create the groups and define what each group is authorised to do on the network devices. The default syntax for the configuration of groups is as follows:



|  |  |
| --- | --- |
| 1  2  3  4  5 | group = name {  [ default service ]  group\_attr  svc  } |

Default services specifies the default <permission> for service authorization, either deny or permit.



|  |  |
| --- | --- |
| 1 | default service = permission |

If omitted, the default is ‘deny’.

**Note: if used, default service must precede all other svc directives in a group clause.**

Group attributes (group\_attr) are attributes that will be inherited by users of the group, these include attributes such as ACL’s and expiry dates.

Services (svc) define services which the group is authorised to execute, these could be services such as telnet or commands that the group is authorised to execute. Authorisation  must  be configured on both the client and the daemon to operate correctly.

Command authorization is configured by specifying a list of <regex> to match command arguments and an action which is a <premission>. The default syntax for command authorization is as follows:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | cmd = string {  permission regex  permission regex  ...  permission  } |

In our example each group will have a default service configured (for network\_admin it is permit while the sys\_admin groups default is to deny all services). Each group will also be associated with the ACL’s that were defined earlier and have an expiry date of 1 January 2015.

The network\_admin team will be authorised to run any command on the devices, while the sys\_admin team will have limited authorisation to run command.

The sys\_admin team will be authorised to run the following commands:

1. Authorised for enable mode.
2. Authorised to run any show commands
3. Authorised to exit (note not end) modes.
4. Authorised to execute configure i.e. configure terminal.
5. Authorised  to  enter Ethernet, FastEthternet or GigabitEthernet
6. interface configuration.
7. Authorised to change the switchport configuration.
8. Authorised to change the description on interfaces.
9. Denied from running any other commands.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51 | # network\_admin group, full access to network devices    group = network\_admin {  default service = permit  expires = "Jan 1 2015"  acl = network\_admin  service = exec {  priv-lvl = 15  }  }    # sys\_admin group, only has read access to the network devices and can change the access vlan on an interface    group = sys\_admin {  default service = deny  expires = "Jan 1 2015"  acl = sys\_admin  service = exec {  priv-lvl = 0  }  cmd = enable {  permit .\*  }  cmd = show {  permit .\*  }  cmd = exit {  permit .\*  }  cmd = configure {  permit .\*  }  cmd = interface {  permit Ethernet.\*  permit FastEthernet.\*  permit GigabitEthernet.\*  }  cmd = switchport {  permit "access vlan.\*"  permit "trunk encapsulation.\*"  permit "mode.\*"  permit "trunk allowed vlan.\*"  }  cmd = description {  permit .\*  }    cmd = no {  permit shutdown  }  } |

###### Users

Once we have configure the groups, we can define the users and associate them with their respective groups. The command syntax for users is the same used for groups. Any commands specifically defined for a user can override those defined for a group.



|  |  |
| --- | --- |
| 1  2  3  4  5 | user = name {  [ default service ]  user\_attr  svc  } |

In the only attributes we will be giving the users are their membership to the groups defined above, passwords and enable passwords.

I will be configuring two user accounts:

1. jonathanm, who will be a member of the network\_admin team and will authenticate from a predefined DES encrypted password.
2. bob, who will be associated to the sys\_admin team and will authenticate from the system /etc/passwd. This requires the user to be configured on the TACACS+ server as well as having a valid password.

To create the password for jonathanm we will be using the tac\_pwd command to generate a DES encrypted password to include in the tac\_plus.conf file:



|  |  |
| --- | --- |
| 1  2  3  4 | root@tacacs:~# tac\_pwd  Password to be encrypted: cisco  6/1aYAL9zcCe.  root@tacacs:~# |

To create the user bob we will have to add the account onto the TACACS+ server:



|  |  |
| --- | --- |
| 1  2  3  4  5 | root@tacacs:~# passwd bob  Enter new UNIX password:test  Retype new UNIX password:test  passwd: password updated successfully  root@tacacs:~# |

Next we will generate enable passwords for each user using two different methods:

1. User jonathanm will have his own enable password associated with his account.
2. User bob will use a a default enable password configured.

The enable password for jonathanm will be generated the same way that his authentication password was defined, using the tac\_pwd command:



|  |  |
| --- | --- |
| 1  2  3  4 | root@tacacs:~# tac\_pwd  Password to be encrypted: 3n@bl3  dBFJQefS4S4Jw  root@tacacs:~# |

For the user bob we will be using one of the specially defined users within the TACACS+ daemon “$enable$”. The “$enable$” user is used for a default, system wide, enable account that is used for any user that does not have a specific enable password configured. We need to generate an enable password for the “$enable$” user using tac\_pwd:



|  |  |
| --- | --- |
| 1  2  3  4 | root@tacacs:~# tac\_pwd  Password to be encrypted: bob  8zFa7qyUEMy6o  root@tacacs:~# |

Now that we have the passwords generated, we can create the user accounts and associate them to their respective groups.  
**Note: we must specify the login account as DES if we are using a password generated by tac\_pwd:**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | # User jonathanm using DES password and enable passwords    user = jonathanm {  member = network\_admin  login = des 6/1aYAL9zcCe.  enable = des dBFJQefS4S4Jw  }    # User bob authenticating from the system /etc/passwd and the default enable password    user = bob {  login = file /etc/passwd  member = sys\_admin  }    # Global enable level 15 password    user = $enab15$ {  login = des 8zFa7qyUEMy6o  } |

Now that we have all the required defaults, user and group parameters we can add the config to the file:



|  |  |
| --- | --- |
| 1 | vi /etc/tacacs/tac\_plus.conf |

Template:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | # Encryption key    key = "tac\_test"    # Set where to send accounting records    accounting syslog;  accounting file = /var/log/tac\_plus/tac\_plus.acct    # ACL for network\_admin group |



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96 | # Encryption key    key = "tac\_test"    # Set where to send accounting records    accounting syslog;  accounting file = /var/log/tac\_plus/tac\_plus.acct    # ACL for network\_admin group    acl = network\_admin {  # allow access from all sources  permit = .\*  # implicit deny (ie: anything else)  }    # ACL for sys\_admin group    acl = sys\_admin {  # allow access from 10.10.10.250 only  permit = ^10\.10\.10\.2$  # implicit deny (ie: anything else)  }    # network\_admin group, full access to network devices    group = network\_admin {          default service = permit  expires = "Jan 1 2015"  acl = network\_admin          service = exec {           priv-lvl = 15          }  }    # sys\_admin group, only has read access to the network devices and can change the access vlan on an interface    group = sys\_admin {          default service = deny  expires = "Jan 1 2015"  acl = sys\_admin          service = exec {  priv-lvl = 0  }  cmd = enable {  permit .\*  }  cmd = show {  permit .\*  }  cmd = exit {  permit .\*  }  cmd = configure {  permit .\*  }  cmd = interface {  permit Ethernet.\*  permit FastEthernet.\*  permit GigabitEthernet.\*  }  cmd =  switchport  {  permit "access vlan.\*"  permit "trunk encapsulation.\*"  permit "mode.\*"  permit "trunk allowed vlan.\*"  }  cmd = description {  permit .\*  }    cmd = no {  permit shutdown  }  }  # User jonathanm using DES password and enable passwords    user = jonathanm {  member = network\_admin  login = des 6/1aYAL9zcCe.  enable = des dBFJQefS4S4Jw  }    # User bob authenticating from the system /etc/passwd and the default enable password    user = bob {  login = file /etc/passwd  member = sys\_admin  }    # Global enable level 15 password    user = $enab15$ {  login = des 8zFa7qyUEMy6o  } |

##### Add TACACS+ as a Startup Service

As it stand once we have added the tac\_plus.conf file to /etc/tacacs, we would be able to start the TACACS+ service and start authenticating and Authorising users. However if for some reason the server restarted we would have to login and start TACACS+ service manually.

The steps bellow will detail adding the TACACS+ service to etc/init.d and /etc/init/rc-sysinit.conf so that it will start automatically upon reboot of the server.

###### /etc/default/tac\_plus

First we need to setup the default behaviour of the tac\_plus command. In this we will define the default configuration file to use when starting the TACACS+ daemon and any other special operations such as debugging and DNS lookups. For a full list of the operations see man tac\_plus:

To do this we need to create a default profile under /etc/default for tac\_plus:



|  |  |
| --- | --- |
| 1  2  3 | touch /etc/default/tac\_plus  chmod 755 /etc/default/tac\_plus  vi /etc/default/tac\_plus |

Add the following template to /etc/default/tac\_plus:



|  |  |
| --- | --- |
| 1  2 | CONFIG\_FILE="/etc/tacacs/tac\_plus.conf"  OTHER\_OPTS="-d 16 -L" |

###### /etc/init.d/tac\_plus

As you can see we currently don’t have any start/stop scripts configured for tac\_plus within init.d:



|  |  |
| --- | --- |
| 1  2  3 | root@tacacs:/etc/tacacs# ll /etc/init.d/tac\*  ls: cannot access /etc/init.d/tac\*: No such file or directory  root@tacacs:/etc/tacacs# |

So the next step is to add the start/stop script to /etc/init.d/tac\_plus:



|  |  |
| --- | --- |
| 1  2  3 | touch /etc/init.d/tac\_plus  chmod 755 /etc/init.d/tac\_plus  vi /etc/init.d/tac\_plus |

Add the following script to /etc/init.d/tac\_plus:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | #!/bin/sh  #  ### BEGIN INIT INFO  # Provides: tac-plus  # Required-Start: $network  # Required-Stop:  # Default-Start: 2 3 4 5  # Default-Stop: S 0 1 6  # Short-Description: Start tac-plus server.  # Description: Run the tac-plus server listening for  # AAA ( access, acounting and autorization request )  # from routers or RAS (remote access servers) via  # tacacs+ protocol  ### END INIT INFO  PATH=/sbin:/bin:/usr/sbin:/usr/bin  DAEMON=/usr/bin/tac\_plus  NAME=tac\_plus  DESC="Tacacs+ server"  OTHER\_OPTS="-d 256" # Default, if no /etc/default/tac-plus available    CONFIG\_FILE="/etc/tacacs/tac\_plus.conf" # Default, if no /etc/default/tac-plus available    test -f $DAEMON || exit 0  if [ -r /etc/default/tac\_plus ] ; then  . /etc/default/tac\_plus  fi  DAEMON\_OPTS="-C $CONFIG\_FILE $OTHER\_OPTS"  case "$1" in  start)  echo -n "Starting $DESC: "  start-stop-daemon --start --quiet --pidfile /var/run/$NAME.pid --exec $DAEMON -- $DAEMON\_OPTS  echo "$NAME."  ;;  stop)  echo -n "Stopping $DESC: "  start-stop-daemon --stop --quiet --pidfile /var/run/$NAME.pid --exec $DAEMON  echo "$NAME."  ;;  \*)  N=/etc/init.d/$NAME  echo "Usage: $N {start|stop}" >&2  exit 1  ;;  esac  exit 0 |

Now that we have added the /etc/default/tac\_plus and /etc/init.d/tac\_plus scipts we will be able to use /etc/init.d/tac\_plus start and /etc/init.d/tac\_plus stop to start and stop the TACACS+ daemon:



|  |  |
| --- | --- |
| 1  2  3  4  5 | root@tacacs:/etc/tacacs# /etc/init.d/tac\_plus start  Starting Tacacs+ server: tac\_plus.  root@tacacs:/etc/tacacs# /etc/init.d/tac\_plus stop  Stopping Tacacs+ server: tac\_plus.  root@tacacs:/etc/tacacs# |

###### /etc/init/rc-sysinit.conf

Now that we have the init.d script configured we can add the /etc/init.d/tac\_plus script to /etc/init/rc-sysinit.conf so that the init.d script will be executed at start up. We do this using update-rc.d:



|  |  |
| --- | --- |
| 1 | update-rc.d tac\_plus defaults |

The command above tells the system to add the tac\_plusService to the default runlevels at startup:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | root@tacacs:/etc/tacacs# update-rc.d tac\_plus defaults  update-rc.d: warning: default stop runlevel arguments (0 1 6) do not match tac\_plus Default-Stop values (S 0 1 6)  Adding system startup for /etc/init.d/tac\_plus ...     /etc/rc0.d/K20tac\_plus -> ../init.d/tac\_plus     /etc/rc1.d/K20tac\_plus -> ../init.d/tac\_plus     /etc/rc6.d/K20tac\_plus -> ../init.d/tac\_plus     /etc/rc2.d/S20tac\_plus -> ../init.d/tac\_plus     /etc/rc3.d/S20tac\_plus -> ../init.d/tac\_plus     /etc/rc4.d/S20tac\_plus -> ../init.d/tac\_plus     /etc/rc5.d/S20tac\_plus -> ../init.d/tac\_plus  root@tacacs:/etc/tacacs# |

##### Verification that TACACS+ is Running

Now that we have the TACACS+ daemon configured, we need to confirm that it is running. To do this we can use netstat, ps and check the syslog logs to ensure everything started correctly:

First start the TACACS+ daemon using the start/stop script defined in /etc/init.d/tac\_plus:



|  |  |
| --- | --- |
| 1  2  3 | root@tacacs:~# /etc/init.d/tac\_plus start  Starting Tacacs+ server: tac\_plus.  root@tacacs:~# |

Check the output in the syslog log file:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | root@tacacs:~# tail -f /var/log/syslog  Oct 25 16:30:15 tacacs tac\_plus[29469]: Reading config  Oct 25 16:30:15 tacacs tac\_plus[29469]: Version F4.0.4.26 Initialized 1  Oct 25 16:30:15 tacacs tac\_plus[29469]: tac\_plus server F4.0.4.26 starting  Oct 25 16:30:15 tacacs tac\_plus[29470]: Backgrounded  Oct 25 16:30:15 tacacs tac\_plus[29471]: uid=0 euid=0 gid=0 egid=0 s=0 |

Make sure the tac\_plus process is running:



|  |  |
| --- | --- |
| 1  2  3  4 | root@tacacs:~# ps aux | grep tac\_plus  root     29471  0.0  0.0  15520   480 pts/0    S    16:30   0:00 /usr/bin/tac\_plus -C /etc/tacacs/tac\_plus.conf -d 16 -L  root     29473  0.0  0.0  11748   920 pts/0    S+   16:34   0:00 grep --color=auto tac\_plus  root@tacacs:~# |

Ok, cool it looks like everything started up correctly, notice that in the output of ps we can see that the TACACS+ daemon started using the options we defined in /etc/default/tac\_plus.

Next lets make sure that the TACACS+ daemon is listening for connections on port 49 TCP using netstat:



|  |  |
| --- | --- |
| 1  2  3 | root@tacacs:~# netstat -an | grep :49  tcp        0      0 0.0.0.0:49              0.0.0.0:\*               LISTEN  root@tacacs:~# |

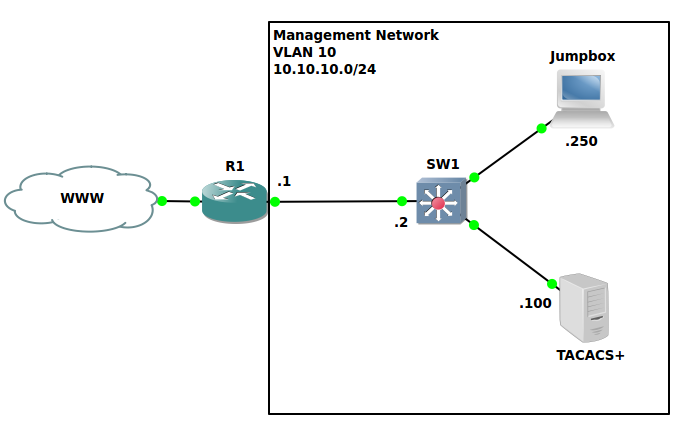
Excellent everything started as it should have and we are listening for connections on TCP port 49.

##### Testing using a Cisco Network Device

To test the TACACS+ server I will be using GNS3 version 1.1, using the following IOU images:

1. R1: i86bi-linux-l3-adventerprisek9-15.2.2.03T.bin
2. SW1: i86bi-linux-l2-ipbasek9-15.1b.bin

The network diagram is as follows:

[](http://www.routingloops.co.uk/wp-content/uploads/2014/10/Selection_028.png)

###### Cisco Device Setup

Firstly we need to define the TACACS+ server we will be using to authenticate and authorise users off of, in this example we are using 10.10.10.100. We also need to specify the encryption key that is configured on the TACACS+ server to ensure that the packets between the client and the TACACS+ daemon are encrypted:



|  |  |
| --- | --- |
| 1  2  3 | tacacs-server host 10.10.10.100  tacacs-server directed-request  tacacs-server key tac\_test |

Next we have to setup the AAA method lists that the Cisco device will use for Authentication, Athorization and Accounting using the following template:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | aaa new-model  aaa authentication login default group tacacs+ local  aaa authentication enable default group tacacs+ enable  aaa authorization config-commands  aaa authorization commands 0 default group tacacs+ local  aaa authorization commands 1 default group tacacs+ local  aaa authorization commands 7 default group tacacs+ local  aaa authorization commands 15 default group tacacs+ local  aaa accounting commands 0 default start-stop group tacacs+  aaa accounting commands 1 default start-stop group tacacs+  aaa accounting commands 7 default start-stop group tacacs+  aaa accounting commands 15 default start-stop group tacacs+  aaa accounting network 0 start-stop group tacacs+  aaa accounting network 15 start-stop group tacacs+  aaa accounting connection 0 start-stop group tacacs+  aaa accounting connection 15 start-stop group tacacs+  aaa session-id common |

1. The **aaa new-model** command enables the AAA security services.
2. **aaa authentication login default group tacacs+ local**: defines the default method list. Incoming ASCII logins on all interfaces (by default) will use TACACS+ for authentication. If no TACACS+ server responds, then the network access server will use the information contained in the local username database for authentication.
3. **aaa authentication enable default group tacacs+ enable**: defines the default list to be used for enable access to the network device via TACACS+ and falling back to the configured local enable password or secret if the TACACS+ server is offline.
4. **aaa authorization config-commands:**ensures that configuration commands are authorised by the TACACS+ server.
5. **aaa authorization commands x default group tacacs+ local**: configures command authorization via TACACS+, falling back to local if the TACACS+ server is offline. We need to configure authorisation for commands run by users in both privilege levels we defined in our TACACS+ groups (i.e. 15 for the network\_admin group and 0 for the sys\_admin group
6. **aaa accounting command x commands 0 default start-stop group tacacs+**: configures command accounting via TACACS+. This must be configured for each privilege level being used on the device.
7. **aaa accounting network x start-stop group tacacs+**: configures network accounting via TACACS+. This must be configured for each privilege level being used on the device.
8. **aaa accounting connection x start-stop group tacacs+**: configures connection accounting via TACACS+. This must be configured for each privilege level being used on the device.

###### Testing Authentication

To test authentication I will log into R1 using both the jonathanm and bob accounts created earlier and display output of the /var/log/tac\_plus logs and /var/log/tac\_plus/tac\_plus.acct logs.

Output from terminal:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | jonathanm@routingloop:~$ telnet 10.10.10.1  Trying 10.10.10.1...  Connected to 10.10.10.1.  Escape character is '^]'.    User Access Verification    Username: jonathanm  Password:    R1>en  Password:  R1# |

Output from /var/log/tac\_plus.log:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | routingloop@tacacs:~$ tail -f /var/log/tac\_plus.log  Sat Oct 25 18:56:00 2014 [1422]: session.peerip is 10.10.10.1  Sat Oct 25 18:56:00 2014 [1580]: connect from 10.10.10.1 [10.10.10.1]  Sat Oct 25 18:56:09 2014 [1580]: cfg\_acl\_check(network\_admin, 10.10.10.1)  Sat Oct 25 18:56:09 2014 [1580]: ip 10.10.10.1 matched permit regex .\* of acl filter network\_admin  Sat Oct 25 18:56:09 2014 [1580]: host ACLs for user 'jonathanm' permit  Sat Oct 25 18:56:09 2014 [1580]: login query for 'jonathanm' tty2 from 10.10.10.1 accepted  Sat Oct 25 18:56:11 2014 [1422]: session.peerip is 10.10.10.1  Sat Oct 25 18:56:11 2014 [1581]: connect from 10.10.10.1 [10.10.10.1]  Sat Oct 25 18:56:11 2014 [1581]: cfg\_acl\_check(network\_admin, 10.10.10.1)  Sat Oct 25 18:56:11 2014 [1581]: ip 10.10.10.1 matched permit regex .\* of acl filter network\_admin  Sat Oct 25 18:56:11 2014 [1581]: host ACLs for user 'jonathanm' permit  Sat Oct 25 18:56:11 2014 [1581]: authorization query for 'jonathanm' tty2 from 10.10.10.1 accepted  Sat Oct 25 18:56:11 2014 [1422]: session.peerip is 10.10.10.1  Sat Oct 25 18:56:11 2014 [1422]: session.peerip is 10.10.10.1  Sat Oct 25 18:56:11 2014 [1582]: connect from 10.10.10.1 [10.10.10.1]  Sat Oct 25 18:56:11 2014 [1583]: connect from 10.10.10.1 [10.10.10.1]  Sat Oct 25 18:56:18 2014 [1582]: enable query for 'jonathanm' tty2 from 10.10.10.1 accepted  Sat Oct 25 18:56:00 2014 [1422]: session.peerip is 10.10.10.1  Sat Oct 25 18:56:00 2014 [1580]: connect from 10.10.10.1 [10.10.10.1]  Sat Oct 25 18:56:09 2014 [1580]: cfg\_acl\_check(network\_admin, 10.10.10.1)  Sat Oct 25 18:56:09 2014 [1580]: ip 10.10.10.1 matched permit regex .\* of acl filter network\_admin  Sat Oct 25 18:56:09 2014 [1580]: host ACLs for user 'jonathanm' permit  Sat Oct 25 18:56:09 2014 [1580]: login query for 'jonathanm' tty2 from 10.10.10.1 accepted  Sat Oct 25 18:56:11 2014 [1422]: session.peerip is 10.10.10.1  Sat Oct 25 18:56:11 2014 [1581]: connect from 10.10.10.1 [10.10.10.1]  Sat Oct 25 18:56:11 2014 [1581]: cfg\_acl\_check(network\_admin, 10.10.10.1)  Sat Oct 25 18:56:11 2014 [1581]: ip 10.10.10.1 matched permit regex .\* of acl filter network\_admin  Sat Oct 25 18:56:11 2014 [1581]: host ACLs for user 'jonathanm' permit  Sat Oct 25 18:56:11 2014 [1581]: authorization query for 'jonathanm' tty2 from 10.10.10.1 accepted  Sat Oct 25 18:56:11 2014 [1422]: session.peerip is 10.10.10.1  Sat Oct 25 18:56:11 2014 [1422]: session.peerip is 10.10.10.1  Sat Oct 25 18:56:11 2014 [1582]: connect from 10.10.10.1 [10.10.10.1]  Sat Oct 25 18:56:11 2014 [1583]: connect from 10.10.10.1 [10.10.10.1]  Sat Oct 25 18:56:18 2014 [1582]: enable query for 'jonathanm' tty2 from 10.10.10.1 accepted |

From the logs above we can see that authentication for the user jonathanm passed the ACL checks and was allowed to authenticate and enable using the passwords that we created using tac\_pwd.

Output of /var/log/tac\_plus/tac\_plus.acct:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | root@tacacs:/home/routingloop# tail -f /var/log/tac\_plus/tac\_plus.acct  Oct 25 18:41:55 10.10.10.1 jonathanm tty2 192.168.1.12 stop task\_id=3 timezone=GMT service=shell priv-lvl=0 cmd=exit  Oct 25 18:50:33 10.10.10.1 jonathanm tty2 192.168.1.12 stop task\_id=5 timezone=GMT service=shell priv-lvl=0 cmd=enable  Oct 25 18:53:36 10.10.10.1 jonathanm tty2 192.168.1.12 stop task\_id=6 timezone=GMT service=shell priv-lvl=0 cmd=exit  Oct 25 18:56:11 10.10.10.1 jonathanm tty2 192.168.1.12 stop task\_id=8 timezone=GMT service=shell priv-lvl=0 cmd=enable  Oct 25 19:01:05 10.10.10.1 jonathanm tty2 192.168.1.12 stop task\_id=9 timezone=GMT service=shell priv-lvl=15 cmd=show running-config  Oct 25 19:01:08 10.10.10.1 jonathanm tty2 192.168.1.12 stop task\_id=10 timezone=GMT service=shell priv-lvl=15 cmd=configure terminal  Oct 25 19:01:11 10.10.10.1 jonathanm tty2 192.168.1.12 stop task\_id=11 timezone=GMT service=shell priv-lvl=0 cmd=exit  Oct 25 19:08:07 10.10.10.1 jonathanm tty2 192.168.1.12 stop task\_id=12 timezone=GMT service=shell priv-lvl=1 cmd=show ip route |

As you can see each command that was issued by the user jonathanm is logged in /var/log/tac\_plus/tac\_plus.acct.

Next, lets try and log in to R1 using the second account we created. We should not be able to authenticate using the username bob as the ACL defined earlier only allows access to the switch on the network (10.10.10.2). The output is from the terminal and /var/log/tac\_plus.log:

Output from terminal:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | jonathanm@routingloop:~$ telnet 10.10.10.1  Trying 10.10.10.1...  Connected to 10.10.10.1.  Escape character is '^]'.    User Access Verification    Username: bob  Password:    % Authentication failed |

Output from /var/log/tac\_plus.log:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | routingloop@tacacs:~$ tail -f /var/log/tac\_plus.log  Sat Oct 25 19:17:42 2014 [1422]: session.peerip is 10.10.10.1  Sat Oct 25 19:17:42 2014 [1619]: connect from 10.10.10.1 [10.10.10.1]  Sat Oct 25 19:17:47 2014 [1619]: cfg\_acl\_check(sys\_admin, 10.10.10.1)  Sat Oct 25 19:17:47 2014 [1619]: ip 10.10.10.1 did not match in acl filter sys\_admin  Sat Oct 25 19:17:47 2014 [1619]: host ACLs for user 'bob' deny  Sat Oct 25 19:17:47 2014 [1619]: login query for 'bob' tty2 from 10.10.10.1 rejected  Sat Oct 25 19:17:47 2014 [1619]: login failure: bob 10.10.10.1 (10.10.10.1) tty2  Sat Oct 25 19:17:51 2014 [1422]: session.peerip is 10.10.10.1  Sat Oct 25 19:17:51 2014 [1620]: connect from 10.10.10.1 [10.10.10.1]  Sat Oct 25 19:18:22 2014 [1620]: 10.10.10.1 tty2: fd 2 eof (connection closed)  Sat Oct 25 19:18:22 2014 [1620]: Read -1 bytes from 10.10.10.1 tty2, expecting 12  Sat Oct 25 19:18:22 2014 [1620]: Error 10.10.10.1 tty2: Null reply packet, expecting CONTINUE |

Perfect, the ACL blocked the user bob from accessing the router as we configured it to.

Now lets see if user bob can log in to 10.10.10.2 and issue some commands, as before the output is from the terminal, /var/log/tac\_plus.log and /var/log/tac\_plus/tac\_pluss.acct:

Output from the terminal:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | routingloop@routingloop-lubuntu:~$ telnet 10.10.10.2  Trying 10.10.10.2...  Connected to 10.10.10.2.  Escape character is '^]'.    User Access Verification    Username: bob  Password:    SW1>en  Password:  SW1#sh ip int brief | in 10.10.10.2  Vlan10                 10.10.10.2      YES manual up                    up  SW1#sh ip route  Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP         D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area         N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2         E1 - OSPF external type 1, E2 - OSPF external type 2         i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2         ia - IS-IS inter area, \* - candidate default, U - per-user static route         o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP         + - replicated route, % - next hop override    Gateway of last resort is not set          1.0.0.0/32 is subnetted, 1 subnets  S        1.1.1.1 is directly connected, Null0        10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks  S        10.1.1.2/32 is directly connected, Null0  C        10.10.10.0/24 is directly connected, Vlan10  L        10.10.10.2/32 is directly connected, Vlan10  O     192.168.1.0/24 [110/11] via 10.10.10.1, 00:47:08, Vlan10  SW1#conf t  Enter configuration commands, one per line.  End with CNTL/Z.  SW1(config)#ip route 1.1.1.1 255.255.255.255 null 0  Command authorization failed.    SW1(config)#inter  SW1(config)#interface ethernet 0/0  SW1(config-if)#switchport mode access  SW1(config-if)#switchport access vlan 2  % Access VLAN does not exist. Creating vlan 2    SW1(config-if)#exit  SW1(config)#exit  SW1#wr  Command authorization failed.    SW1#reload  Command authorization failed.    SW1# |

Output from /var/log/tac\_plus.log:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | root@tacacs:/home/routingloop# cat /var/log/tac\_plus.log | grep authorization  Sat Oct 25 20:51:25 2014 [1960]: authorization query for 'bob' tty2 from 10.10.10.2 accepted  Sat Oct 25 20:51:51 2014 [1963]: authorization query for 'bob' tty2 from 10.10.10.2 accepted  Sat Oct 25 20:51:57 2014 [1965]: authorization query for 'bob' tty2 from 10.10.10.2 accepted  Sat Oct 25 20:51:59 2014 [1967]: authorization query for 'bob' tty2 from 10.10.10.2 accepted  Sat Oct 25 20:52:12 2014 [1969]: authorization query for 'bob' tty2 from 10.10.10.2 rejected  Sat Oct 25 20:52:21 2014 [1970]: authorization query for 'bob' tty2 from 10.10.10.2 accepted  Sat Oct 25 20:52:27 2014 [1972]: authorization query for 'bob' tty2 from 10.10.10.2 accepted  Sat Oct 25 20:52:32 2014 [1974]: authorization query for 'bob' tty2 from 10.10.10.2 accepted  Sat Oct 25 20:52:37 2014 [1976]: authorization query for 'bob' tty2 from 10.10.10.2 accepted  Sat Oct 25 20:52:38 2014 [1978]: authorization query for 'bob' tty2 from 10.10.10.2 accepted  Sat Oct 25 20:52:40 2014 [1980]: authorization query for 'bob' tty2 from 10.10.10.2 rejected  Sat Oct 25 20:52:47 2014 [1981]: authorization query for 'bob' tty2 from 10.10.10.2 rejected  root@tacacs:/home/routingloop# |

As you can see from the logs above, any commands that bob was not authorised to execute have been rejected.

Output from /var/log/tac\_plus/tac\_plus.acct:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | root@tacacs:/home/routingloop# tail -f /var/log/tac\_plus/tac\_plus.acct  Oct 25 20:51:25 10.10.10.2 bob tty2 10.10.10.250 stop task\_id=55 timezone=GMT service=shell priv-lvl=0 cmd=enable  Oct 25 20:51:51 10.10.10.2 bob tty2 10.10.10.250 stop task\_id=56 timezone=GMT service=shell priv-lvl=1 cmd=show ip interface brief  Oct 25 20:51:57 10.10.10.2 bob tty2 10.10.10.250 stop task\_id=57 timezone=GMT service=shell priv-lvl=1 cmd=show ip route  Oct 25 20:51:59 10.10.10.2 bob tty2 10.10.10.250 stop task\_id=58 timezone=GMT service=shell priv-lvl=15 cmd=configure terminal  Oct 25 20:52:21 10.10.10.2 bob tty2 10.10.10.250 stop task\_id=59 timezone=GMT service=shell priv-lvl=15 cmd=interface Ethernet 0/0  Oct 25 20:52:27 10.10.10.2 bob tty2 10.10.10.250 stop task\_id=60 timezone=GMT service=shell priv-lvl=15 cmd=switchport mode access  Oct 25 20:52:33 10.10.10.2 bob tty2 10.10.10.250 stop task\_id=61 timezone=GMT service=shell priv-lvl=15 cmd=switchport access vlan 2  Oct 25 20:52:37 10.10.10.2 bob tty2 10.10.10.250 stop task\_id=62 timezone=GMT service=shell priv-lvl=0 cmd=exit  Oct 25 20:52:38 10.10.10.2 bob tty2 10.10.10.250 stop task\_id=63 timezone=GMT service=shell priv-lvl=0 cmd=exit |

The accounting logs show the commands that bob was authorised to execute.

So after seeing the results of the above test we can now say that our TACACS+ server is correctly configured to provide AAA services to our network devices. I hope this has been helpful, and please add to the comments if you have any questions.

References: