

Tacacs lab

getting the tacacs+ server configured

```
$ sudo apt-get install tacacs+
$ sudo groupadd -r cisco
$ sudo vi /etc/tacacs+/tac_plus.conf
```

change the following settings

- we want to set the shared key for routers who want to use our service to TacacsPassword
- We also want to limit access for users based on groups. For this example we will use settings in tac_plus.conf

change this line

```
key = TacacsPassword
```

In the real world we'd choose a much stronger shared key e.g.

```
$ apt-get install pwgen
$ pwgen -s 64 1
BR5WUWgJLkuxyqfmwfrlRC8JW54bpm3a2rMEe1IWwwpupwGBreGCXGTdbqkMGo2F
```

... then at the end of the file add:

```
# "level 2" users who cannot "debug" or "config"
#
group = l2_tacacs_users {
    default service = permit
    login = file /etc/passwd
    enable = file /etc/passwd
    service = exec {
        priv-lvl = 15
    }
    cmd = configure {
        deny "."
    }
    cmd = debug {
        deny "."
    }
}
```

Users who can only run specified commands

```
group = field_Techs { default service = deny service = exec { priv-lvl = 2 }  
cmd = enable { permit . } cmd = show { permit . } cmd = do { permit .* } }
```

```
group = netops {  
    default service = permit  
    login = file /etc/passwd  
    enable = file /etc/passwd  
    service = exec {  
        priv-lvl = 15  
    }  
}
```

```
#  
# group member not in password file  
# use 'tac_pwd -m' command to encode password  
# this generates an MD5 password but it's still  
# labelled 'des' to show it's not cleartext  
#  
user = training {  
    member = l2_tacacs_users  
    login = des $1$K/$yjCG7R0iaM9I7ze.BBnxk0  
}  
  
    user = techie {  
        member = field_Techs  
        login = des $1$K/$yjCG7R0iaM9I7ze.BBnxk0  
    }  
user = admin {  
    member = netops  
    login = des $1$K/$yjCG7R0iaM9I7ze.BBnxk0  
}
```

```
#  
# "level 2" users with full privileges  
#
```

```
#  
# group member with entry in password file  
# delete user here to deny access to devices  
#  
user = sysadm {  
    member = netops
```

```
}
```

check tacacs_plus config

```
$ sudo service tacacs_plus check
```

You should see a response like:

```
* Checking TACACS+ authentication daemon configuration files successful tacacs+
```

restart tacacs_plus to pick up the new settings

```
$ sudo service tacacs_plus restart
```

getting a cisco device to talk to your tacacs

Enter configuration mode:

```
tacacs-server host 10.10.0.x
tacacs-server key TacacsPassword
```

(Later versions of IOS (15...) have an alternative mechanism for defining these parameters but this can be used on all systems for now.)

Check that you can reach the tacacs server and authenticate correctly:

```
test aaa group tacacs+ sysadm <password> port 49 legacy
```

You should see a response like:

```
Attempting authentication test to server-group tacacs+ using tacacs+
User was successfully authenticated.
```

Now you can finish configuring the router to use tacacs for login control:

```
aaa new-model
```

```
aaa authentication login default group tacacs+ enable
aaa authentication login NSRCCONSOLE local-case
```

```
aaa authentication enable default group tacacs+ enable
aaa authorization exec default group tacacs+ none
aaa accounting delay-start
aaa accounting exec default start-stop group tacacs+
aaa accounting commands 15 default start-stop group tacacs+
```

```
! This lets us login via the console even if tacacs isn't working
username NSRCCONSOLE password 0 tpyPo9dT
line con 0
  exec-timeout 15 0
  login authentication NSRCCONSOLE
```

Now you can verify accounting

```
Router#show aaa sessions
Router#show aaa user all
```